Sun Technical Bulletin

February 1993

Technical Information Services

Sun Microsystems Computer Corporation
A Sun Microsystems, Inc. Business

2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

Part No: 812-9302-01
Issue 1993 - 02

The *Sun Technical Bulletin* is distributed to customers with software/hardware or software-only support contracts. Send comments or corrections by electronic mail to *stb-editor@Sun.COM*, or by mail to:

Sun Technical Bulletin
Sun Microsystems Computer Corporation
2550 Garcia Avenue , M/S MPK02-04
Mountain View, CA 94043-1100 U.S.A.

Contents

1. STB Editor's Introduction 1-1

STB Overview 1-1

2. Solaris 2-1

Federated Services 2-1

Introduction 2-1

A Vision for the '90s and Beyond 2-2

Solaris Federated Services Architecture 2-5

Concurrent Initiatives 2-16

3. Network 3-1

The State(lessness) of Being: An NFS Update. 3-1

Introduction 3-1

Performance 3-2

Naming and Administration. 3-4

Consistency 3-7

Scalability 3-8

Availability 3-11
Security 3-13
What Next? 3-14
Network Load: X-terminals vs. Workstations 3-15

4. SPARC. 4-1
SRMMU, TLBs, and Contexts 4-1
SPARC Architecture 4-1
Architecture: Implementation of Hardware and Software 4-2
The SPARC Reference MMU 4-3
Contexts 4-5
Address Translation 4-6
Translation Lookaside Buffer (TLB) 4-9

5. Software Release Versions. 5-1
Current Sun Software Products and Release Versions 5-1
Operating Systems 5-1
Window Systems & Desktop Utilities 5-1
Graphics Software 5-2
Software Development 5-2
System Management 5-3
Other Products 5-3
Connectivity Products 5-4
Updated Sun Operating System Support Matrix 5-5

6. Notes & Comments **6-1**
World Hotlines 6-1
Reporting Bugs 6-3
Submitting Software Bugs: United States and Canada . . 6-3
Submitting Software Bugs: Europe 6-4
Submitting Software Bugs: Intercon. 6-8
Your STB Subscription 6-10
Index

STB Editor's Introduction 1

STB Overview

This month's STB contains articles of general interest in the topics outlined below.

Solaris

"Federated Services" is a reprint of a SunSoft technical white paper, describing the various services available in SonSoft's distributed computing strategy. The Solaris Federated Services architecture extends the distributed computing foundation to enable users to access resources across a heterogeneous network supporting services from many different vendors. Solaris can now link multiple autonomous distributed services in "plug and play" fashion, allowing them to co-exist and cooperate on the network in a manageable and consistent way. This article articulates the Federated Services approach and details the concurrent activities underway by SunSoft and its partners.

Network

"The State(lessness) of Being: An NFS Update, " by SMCC's Hal Stern, was originally published, in much shorter form, in Hal's /sys.admin column in *SunWorld* magazine. This month, we present the entire, unabridged version of the article, which describes how system enhancements can dramatically improve NFS performance and reliability in a distributed computing environment. The article explores various NFS-related research and development efforts of recent years, and what they imply about NFS performance, administration, scalability, security, etc.

We also include “Network Load: X-terminals vs. Workstations,” by SMCC Technical Product Marketing personnel. This is a brief summary of a recent study comparing the network overhead of workstations and X-terminals.

SPARC

“SRMMU, TLBs, and Contexts” is an overview of virtual-to-physical address translation on a SPARC system. The article discusses the low-level architecture and implementation of virtual memory on SPARC, concentrating on the SPARC Reference Memory Management Unit (SRMMU), the Translation Lookaside Buffer (TLB), and the concept of a process’s context.

Software Release Versions

This section includes the current Sun software products and release version tables. We also include a revised updating of Sun’s support matrix for Operating System release levels and platforms.

Notes & Comments

World Hotlines

For Sun customers world-wide served by your local service groups, use the customer service telephone numbers listed in this monthly item. Also, look to this section during the upcoming year for details on your local support call policies and procedures.

Reporting Bugs

The information in this monthly section continues to be expanded as Sun software service centers are added world-wide. The section includes a list of Sun service centers, addresses, e-mail hotlines, telephone hotlines, and FAX numbers.

Your STB Subscription

This notice is published monthly, giving customers useful information regarding ordering multiple STB hardcopy subscriptions, and duplicating additional STB copies from a single “master” copy. This duplication permission is limited, as detailed in the note.

We would like to encourage our readership to e-mail us any comments or suggestions regarding the content of the Sun Technical Bulletin, including requests for articles on specific subject matters. We are aiming at a broad spectrum of topics that are of interest to all sectors of our audience, both highly experienced technical people and relatively inexperienced users who are new to desktop workstations. We are not the appropriate channel for researching any and all technical problems that may come up in the field, but we will do our best to follow up on questions about articles that we have printed in the recent past.

Please let us know what you think, and we will do our best to accommodate you in future issues.

Thanks.

The STB Editor
sun!stb-editor or *stb-editor@Sun.COM*

Solaris 2

Federated Services

The following article is a reprint of a SunSoft technical white paper.

Introduction

This paper articulates SunSoft's vision for distributed computing in the 1990s and beyond. The Solaris Federated Services architecture, which is a key component for accomplishing this vision, is also discussed.

The SunSoft vision of distributed computing is to make computing with Solaris global, mobile, and simple. This means worldwide networks with mobile users, environments, and systems that are easy to administer. A key requirement for such an environment is the ability for users to access resources across a heterogeneous network supporting distributed services from many different vendors. For example, naming, filing, and security services for NetWare, ONC+, DCE, OSI, and others should be able to coexist neatly. Federated Services extends the Solaris distributed computing foundation to address this requirement. It provides Solaris with the ability to link multiple autonomous distributed services — such as multivendor filing, naming, and security services — in "plug and play" fashion and allows these services to coexist and cooperate on the network in a manageable, consistent way.

The Solaris operating environment provides standardized Federated Service Interfaces (FSI), which allow vendors to add their distributed services to Solaris in a well-integrated manner, such that these services become peers to the ONC+ core services within Solaris. New and existing applications take advantage of these multivendor distributed services using the consistent Solaris Application Programming Interface (API).

The Solaris Federated Services architecture is an important dimension in evolving Solaris to support the vision of global, mobile, and simple computing. A range of concurrent initiatives are underway by SunSoft and its partners in support of this vision.

A Vision for the '90s and Beyond

The last ten years have seen a revolution in information technology. Users who formerly shared a mainframe or minicomputer with hundreds of other users are now accustomed to controlling the same power on their desktops. In just a few years, most enterprises have become complex webs of computers of various sizes, attempting (and often failing) to interoperate through a web of networking technologies. Gone are the days when users waited months for their MIS department to create a simple report format. Now users expect to access and manipulate data throughout the corporation. Information technology vendors are quick to offer solutions for the sort of global distributed computing that users require. They have been slower to address the more difficult problem of making all of these solutions work together well on one network.

SunSoft is well positioned to meet the challenges of distributed computing in the next decade. As part of Sun Microsystems Inc., SunSoft pioneered interoperability in mixed-vendor networks and is widely regarded as a leader in open systems. Sun's ONC NFS distributed file system technology has won industry-wide acceptance as a de facto standard for file sharing in heterogeneous environments, with an installed base exceeding 3.1 million nodes in early 1992. Now, SunSoft is dedicated to evolving its Solaris operating environment to meet the distributed computing needs of information processing users in the 1990s.

The SunSoft vision of distributed computing is to make computing with Solaris global, mobile, and simple:

- Global — Users have access to resources such as printers, servers, applications, and files, across a heterogenous, global network, even extending beyond their own enterprises.

- Mobile — Users move about independently across the enterprise while maintaining consistent access to their default environments and network services. Systems can be disconnected from the network, used productively and easily reconnected back to the network.
- Simple — Administrator resources are utilized efficiently through the use of automated system and network administration. Cost of ownership is further improved through reducing costs associated with incorporating new technologies as they become available and minimizing end-user training.

Global Access to Resources

Global access to resources is a requirement in today's business environments. *Global* refers to the ability to enlist services worldwide across a large enterprise network, or a network that crosses enterprises. Global access includes reading and writing files, directories, and databases, independent of where they're physically located. It also means that applications are available across a large network, as are devices, such as printers at remote locations.

Solaris distributed computing offers users transparent access to a global network of resources. The resources are at the users' fingertips, independent of the location of the users or the resources, blurring the concept of where the physical network begins and ends. Users focus on getting their work done and not on understanding how to navigate the network. As the global network changes, the network adapts to change — some resources become unavailable, while others are added to the network, but little or no intervention is required by the user.

For example, a company licenses new revisions of Solaris and their favorite spreadsheet program. The system administrator installs the new software on one "master server" and the update propagates throughout the network to all affected users transparently. Users do not need to reinstall. The next time they use the application, the new software is already there.

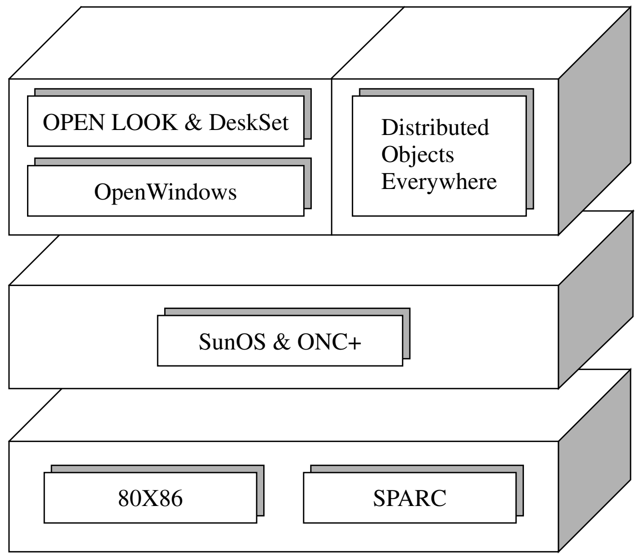

Figure 2-1 Solaris Operating Environment

Mobile Users, Systems, and Environments

Technology has become more sophisticated in the past decade, and workers are finding that their jobs are becoming more sophisticated too. The norm has become moving about rather than remaining stationary in a single office. Many employees work at their homes in the evenings or periodically in the daytime. Others travel constantly. Still others spend a great deal of time visiting remote offices. All these workers need access to their computing resources from wherever they are. In addition, system administrators need to be able to move resources around their networks to meet the dynamic requirements of a world where successful companies reorganize frequently.

In the SunSoft vision, users travel to remote offices, log into an available workstation, and find their home computing environment — the tools they use and the services they need are all there. Users work on portable systems away from their offices, and still their home environments are with them. In one example, the user plugs the portable into a phone jack and within minutes is working with refreshed data from the network. The system has automatically updated the user’s files on the home server and has downloaded incoming electronic mail and other data to the portable’s local cache. No commands are entered; no login procedure is followed; a simple notification to refresh and a simple plug into the wall are all that's required to continue working on the portable. In other examples, users are working on wireless systems and do not need refreshes at all.

Simple Distributed Computing

Simple focuses on lowering the cost of ownership for systems running Solaris, particularly in the area of system and network administration. Simple distributed computing means simplicity of installation, reconfiguration, maintenance, and upgrades, with the goal being to reduce administration to just authorization. It means greater transparency of mobile, global features and the services that enable them. *Simple* also pertains to removing the cost of encumbrances to the use of new distributed services as they become available and enabling the development of innovative applications for administration of these services and physical resources in global, heterogeneous networks.

For example, access to services across a worldwide heterogeneous network requires that a particular user's environment support a wide range of services. Some sites may be using Solaris's core Open Network Computing Plus (ONC+) services; others may be using Novell NetWare or Open Software Foundation's Distributed Computing Environment (DCE). With the realization of the "simple" component of the vision, each of these services easily plugs into Solaris. For the network administrator, adding or replacing services is as simple as setting a flag and consistent object-oriented tools automate administration tasks, such as adding users, server, or printers to the heterogeneous network.

For end users and software developers, access to network resources, such as files, mail boxes, printers, and hosts is completely transparent. They never need to know which specific service is actually used to access the resources. Addition of new resources to the network does not require re-engineering of existing applications.

Solaris Federated Services Architecture

Federated Services provides Solaris with the ability to support multiple autonomous distributed services in "plug and play" fashion and allows these services to coexist and cooperate on the network in a manageable, consistent way. Distributed services, such as filing, naming, and security services, provide the basis for transparent operation and efficient administration of a client/server environment. A key requirement of global, mobile, and simple computing is support for heterogeneity, specifically for multivendor distributed services, such as naming, filing, and security services. Federated services extends the Solaris distributed computing foundation to address this requirement in a well-integrated manner. Federated services has the following benefits for Solaris end-users, application developers, and administrators:

- Users are empowered by being able to transparently access resources, such as files, servers, and applications, across multivendor global networks.
- Developers build applications that are shielded from the distributed services they use and transparently operate in heterogenous network environments, allowing portability and extension of scope.
- Administrators easily add or replace distributed services, like distributed file systems, without requiring modification of applications, allowing simplified support of heterogenous environments and migration to new technologies.

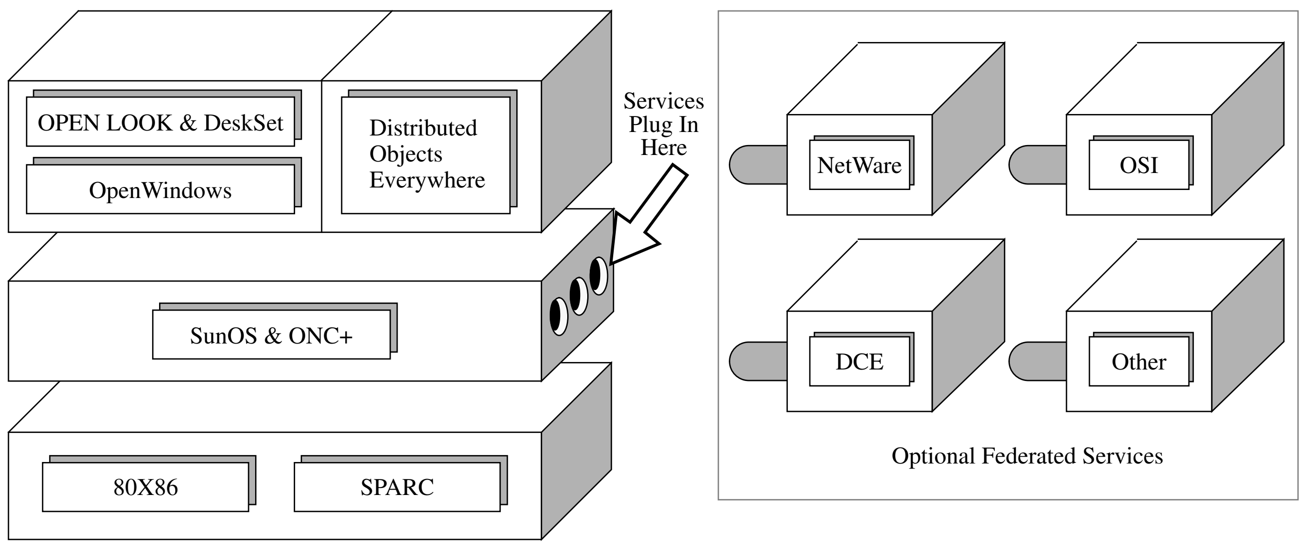

Figure 2-2 Solaris Federated Services Architecture

The Solaris Federated Services framework is an important dimension to enabling SunSoft's vision of global, mobile, and simple distributed computing. Solaris applications and users can easily access multivendor resources across global networks. Federated Services promotes mobility of users and systems as application and system services are shielded from the underlying system services in use across the enterprise. Further, it supports the goal of low cost of ownership by removing encumbrances to support of heterogeneous distributed computing technologies and incorporating new innovative distributed services.

Federated Services is based on some key assumptions:

- Heterogenous computing is the rule. No single distributed service (for example, file service or naming system) will predominate. Users must use a variety of protocols and services across the enterprise.
- Even if a particular service is "best" today, something better may come along tomorrow. Users will require an operating environment that lets them adopt innovative technologies quickly and easily.

Solaris provides standardized Federated Service Interfaces (FSI), allowing the tight integration of ONC+ core services, optional filing, naming, and security services from different vendors within Solaris. Existing and new applications take advantage of these multivendor distributed services using the consistent Solaris API (Application Programming Interface). In this way, Solaris is designed for evolution rather than revolution. Solaris customers will not be forced to abandon their investment in existing applications and equipment to take advantage of new technologies.

Solaris is a vehicle for interoperability. Its architecture enables support of existing and emerging industry standards, ensuring that Solaris applications will enjoy broad usability within multivendor networks. SunSoft is committed to providing full integration of the most widely used protocol stacks and distributed computing facilities in the industry. Users are given the freedom to choose among the growing number of technology alternatives that are available. Yet Solaris controls complexity through the use of intelligent interfaces which allow well-integrated support of multivendor distributed computing services. The next several examples show how Federated Services promote SunSoft's distributed computing vision.

Federated Filing

File Systems: Fundamental Applications

Working with files is one of the most basic, yet fundamentally important tasks that users perform using computers. Traditionally, the "applications" that allow users to accomplish this — referred to as *file systems* — have been available as components of operating systems. New functionality and innovation have come from operating system vendors. For the most part, this model has served users well. However, the model has introduced many different file system types for many different operating systems. Additionally, file system users have not benefitted as greatly from the rapid innovation that independent software vendors can bring to the marketplace. In 1984, Sun Microsystems developed a concept called the "vnode interface" to permit integrated support of multivendor file systems within Solaris and promote innovation among software vendors. Federated filing involves extending this vnode interface to address the global, mobile, simple vision, making it easier for users to access files across a heterogeneous, multi-protocol environment.

Vnode Interface: Multivendor File System Support

The vnode interface permits an operating system to support a variety of file system types by defining interfaces between file system implementations and the rest of the kernel. Examples for Solaris have included ONC NFS and RFS for distributed file access, the UNIX File System (UFS) for local file access, and a number of special-purpose file systems. This capability has long been available in Solaris, and the vnode interface has become a standard way of supporting multivendor file systems. Support for multivendor file systems enables Solaris users to interoperate in heterogeneous environments and to take advantage of special-purpose file systems.

Over time, the vnode interface has evolved to accommodate advances in technology, such as new virtual memory systems. This evolution has occurred through an ongoing process that has made the vnode interface good at handling all of the various new technologies, but not as good at accommodating today's total computing environments. In short, the vnode interface has begun to carry with it some overhead to compensate for its flexibility. Federated filing extends the vnode interface to make Solaris more robust in adapting to future file system technologies.

Federated Filing: More Rapid Innovation

With federated filing, the Solaris vnode interface is enhanced with a feature similar to the AT&T System V Release 4 (SVR4) STREAMS framework, which allows efficient integration of multiple protocol stacks, namely the ability to stack and unstack file system modules. In addition to allowing new file systems to be added to Solaris more easily, it permits the addition of features common to several file system types, such as caching. It also permits runtime alteration of file system behavior.

Federated filing in Solaris allows for greater innovation by SunSoft as well as by independent software vendors. It is expected that new file systems such as DCE DFS will easily integrate with Solaris by taking advantage of the enhanced vnode interface. All new file systems are accessed by users and applications using the same high-level interfaces. For example, applications use `open`, `read`, and `write` to access files regardless of which file system is used. Users benefit from more rapid availability of new functionality, such as performance enhancements, tools for system administrators, support for additional transports, and other new features. System software developers can easily build file system products that work well with Solaris, without spending countless hours reviewing operating system source code. System administrators can pick and choose between alternative distributed file systems in the marketplace based on the needs of their organizations.

Federated filing is a major step toward SunSoft's vision for global, mobile, simple computing with Solaris. It supports mobile computing because new functionality can easily be layered on top of file systems, allowing specialized applications to be developed for portable systems and for those users moving from office to office. For global access, support of multivendor file systems allows Solaris users to access files through whatever service is being used at the physical location of the files, regardless of the services being used locally by the user. Finally, simple distributed computing is provided by consistent interfaces for all file systems, enabling applications to take advantage of many different file systems without modification or re-compilation.

Federated Naming

Naming Systems: Pervasive Throughout Distributed Systems

Naming systems are critical for transparently accessing and administering entities in global networks. By *entities*, we mean any resource stored on the network, such as applications, files, peripherals, users, and in the future, the "objects" that will be the building blocks of distributed applications. The main function of a naming system is to maintain a mapping of names with network entities and to allow applications and users to access these entities.

Historically, each type of network entity has had its own naming system. For example, services such as ONC Network Information Service Plus (NIS+) include integrated naming systems for resources such as hosts and users. The file systems used by various operating systems like UNIX and VMS also have integrated naming systems defining syntax and access commands. And finally, productivity applications such as stock quotation and personal calendar managers also typically have integrated naming systems.

A major impediment to the goal of global, mobile, simple computing, therefore, is the diversity and incompatibility of naming systems. These incompatibilities prevent a user or application from being able to easily access entities around the global network, portions of which may be served by different name services. Solaris includes capabilities allowing users and applications to easily and consistently access the full range of heterogeneous resources throughout the global network. Solaris will permit integration of services, such as NIS+, the OSI X.500 Directory Services and Open Software Foundation's Cell Directory Service (CDS) as well as specialized naming systems supporting system resources and applications.

Federated Naming: Simple Access to Diverse Range of Network Resources

Federated naming in Solaris allows integrated and consistent support within Solaris of multivendor naming systems, including global and enterprise name services supporting high-level entities such as hosts and users and specialized naming systems for files, printers, spreadsheets and personal calendars. Federated naming is the way to allow Solaris applications to consistently and easily access this diverse range of entities.

Not only does federated naming allow Solaris to support a range of naming systems, it also specifies a federated naming service interface allowing these naming systems to cooperate together to implement a consistent application programming interface for resource access across the global network. This naming system cooperation aspect of federated naming allows Solaris to support and applications to use *composite names* formed by combining names from different naming systems in the federation. Thus, Solaris applications will be able to easily traverse the global heterogeneous network to access network entities.

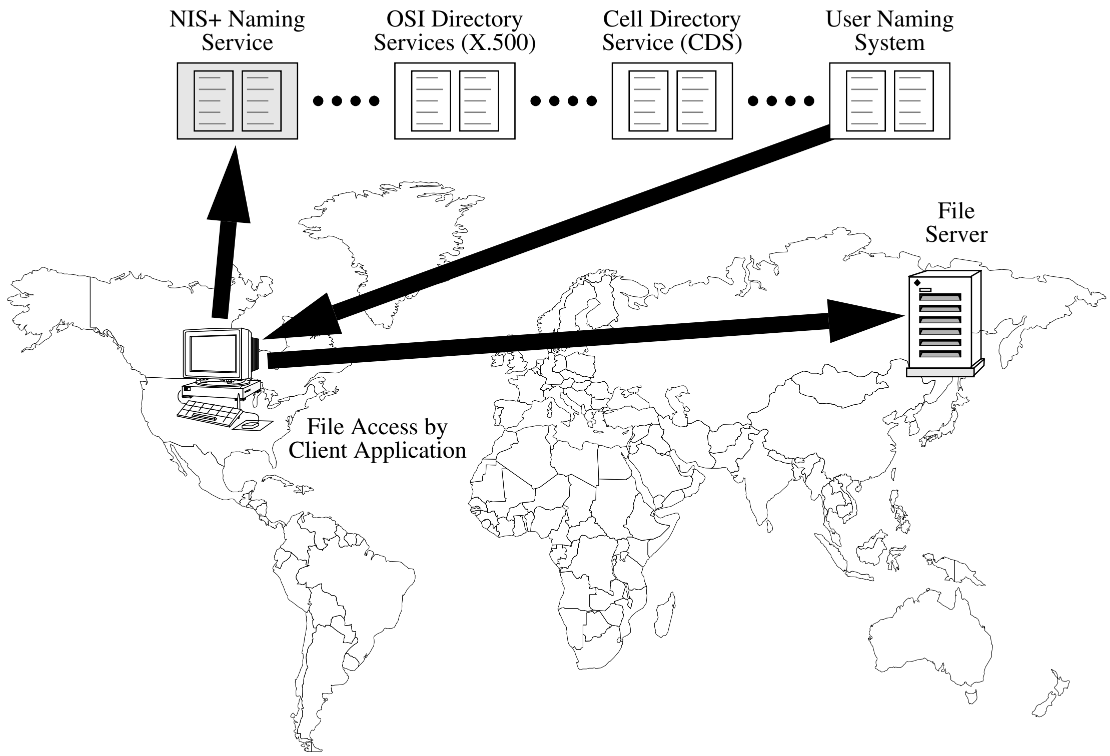

Figure 2-3 Federated Naming Across Global Networks

Federated naming will provide benefits to developers, administrators, and end-users alike. Developers benefit by taking advantage of the consistency of the naming interface provided by the federated naming system. The federated naming service enables simple access by applications to all types of heterogeneous resources across the global network through a single naming interface, allowing simplified programming of distributed applications. Clients of the federated naming service can resolve names without knowledge of the syntax of the names or specific naming interfaces, as applications treat resource names as opaque entities. The addition of new types of naming systems, to support existing or new network entities, such as distributed objects, does not require changes to clients. The federated naming service off-loads the name resolution function from clients, permitting extensibility and lower cost-of-ownership.

Clients of the federated naming service, such as Solaris users and applications, will also benefit by using intuitive and consistent methods, such as user, group, or organization-centric names, as opposed to the traditional host-centric naming approach, for accessing network resources. Use of host-independent names for network resources also promotes mobility of users and systems.

Consistent support for naming of and access to multivendor resources permits simplified administration of a global heterogeneous network. In addition, the service is extensible in allowing support for new types of administrative objects, holding promise for significant improvements in cost of ownership.

SunSoft is working with system and application software vendors in getting global and enterprise-level naming systems, such as X.500, DCE CDS, and naming systems for application-specific entities to support the federated naming architecture. In addition, the federated naming service interface will be published and open, allowing developers to easily incorporate their naming systems into the federated naming architecture.

Flexible Security Schemes

Global, mobile client/server computing requires system security that is comprehensive and pervasive. Allowing users transparent access to enterprise-wide resources can easily conflict with the goal of maintaining the privacy and integrity of workspaces. Flexible security systems are needed to win the confidence of end-users, system administrators, and business executives.

SunSoft is committed to providing complete solutions to the network security problems of authentication and authorization. *Authentication* involves verifying a client's identity for network operations, and *authorization* refers to verifying a client's rights to perform the desired operation. The challenge is not only to provide a reliable mechanism for authentication and authorization, but also to allow the use of multiple authentication and authorization schemes.

Authentication in a distributed systems environment is achieved by the client and server using cryptographic algorithms to demonstrate that they know a secret, without revealing the secret. Distributed applications can use the authentication mechanisms to securely establish the identity of clients, as well as to encrypt data between the client and server, assuring the privacy and integrity of data communications.

The Solaris Federated Services architecture supports a variety of authentication technologies, including the industry-standard Diffie-Hellman, Kerberos, and RSA mechanisms. Network services, such as naming and file services, receive and maintain a user's credentials. A generalized internal interface lets services request and verify a user's credentials regardless of the technology being used to carry out the authentication.

Developers can use TI-RPC (Transport Independent Remote Procedure Call) in their distributed applications without having to choose an authentication scheme at the time of development. The system and network administrators can choose the authentication flavor when the application is installed or at runtime. For example, network administrators can select an authentication scheme to be used when configuring a network service; if the Kerberos authentication system is installed on the network, an administrator can configure NFS, NIS+, and other services to use it. Solaris will continue to provide flexible solutions to meet enhanced security needs of mobile, global client/server computing.

Distributed Application Platforms

As users move away from mainframe-based computing toward true client/server computing, we are seeing increased use of distributed applications which can better harness the power of networked resources. A distributed application is one that runs across more than one system on the network, making use of both local and remote compute resources. A typical application has a client "front end," usually employing a graphical user interface (GUI), that accesses one or more servers. These servers may store needed data or provide other compute resources to augment the capability of the application's local CPU.

Remote Procedure Call (RPC)

Remote procedure calls, or RPCs, can be viewed as building blocks for distributed applications. The client portion of an application can invoke the server by issuing an RPC. The RPC routine is called by the client program the same way a local routine is called. The difference is that the processing is done remotely by the server, and the results are then sent back to the client program.

There are many varieties of RPCs in the industry today. Solaris supports a Transport Independent RPC (TI-RPC) mechanism, which is compatible with the widely installed ONC RPC. In addition, Solaris can accommodate a variety of RPC mechanisms, such as OSF DCE or the emerging OSI RPC standard, that can coexist peacefully with each other. This allows integrated support for multivendor distributed services, built using these different RPC technologies. It also gives developers more freedom in designing and implementing RPC-based applications that can interoperate in heterogeneous distributed computing environments.

There are also several higher-level tools available to ease the development of distributed applications. The ONC RPC Application Toolkit provides a code generator that automatically generates client- and server-specific routines. Solaris includes the ToolTalk messaging service for application-to-application messaging. ToolTalk allows applications to register for services and data, and to exchange messages with each other. In either case, these tools for distributed computing rely on the underlying RPC foundation.

New Models of Distributed Computing — Distributed Objects

SunSoft is continuing to offer improvements in technology to support a distributed object computing environment within Solaris. An initiative in the Solaris object computing direction is the Distributed Objects Everywhere (DOE) project, which includes a collection of runtime services that manage the existence and execution of objects as well as interaction between objects. Included is also a single Interface Definition Language (IDL) that will allow developers to create distributed applications that call dynamically on hundreds or thousands of objects regardless of their location and host environment on the network. Such a computing model promises developers unprecedented flexibility in delivering application solutions and offers users the opportunity for a cooperative computing environment.

Insulation from Low-Level Protocols

The Solaris Distributed Computing Framework supports multiple protocol stacks for maximum interoperability and insulation of network services from underlying communication protocols. A particular stack of protocols can be selected at runtime to match the services available on another system. This is true both at the RPC level and for the underlying network transport protocols.

Modularity at Protocol Level

Traditionally, distributed applications, whether using RPCs or the older Berkeley sockets, have been dependent on a specific network transport. The Solaris architecture again uses a standardized interface to shield applications from the underlying low-level protocols. This interface is the System V Transport Layer Interface (TLI), equivalent to the X/Open Transport Interface (XTI).

TLI provides a common means of accessing connection-oriented or connectionless transport services. Addressing is opaque, so applications do not depend on any of the properties of a particular addressing scheme, such as Internet protocol port numbers.

TI-RPC uses TLI for transport independence. That means that a distributed application may be written for TCP, for example, but will run unmodified when the user migrates to an OSI-based network. The users' investment in applications is therefore protected. And software vendors can economically ship shrink-wrapped software, without having to produce multiple versions of their applications to cater to different types of networks.

Kernel Interfaces

A number of standard interfaces and frameworks in the Solaris kernel facilitate the development of new device drivers and protocols. The reduced development costs ensure that a large selection of connectivity products will be available for Solaris.

Developers can gain access to low level protocols through the Transport Provider Interface (TPI) and the Data Link Provider Interface (DLPI). Developers can also provide alternative implementations of these low-level protocols. A protocol implementation offers its services to the system through one of these provider interfaces.

The STREAMS framework provides a consistent and powerful set of basic mechanisms for implementing protocol stacks and device drivers. While Solaris also supports the Berkeley sockets mechanism, STREAMS is the native facility of UNIX SVR4 and of Solaris 2.*x*.

Implementors of transport layer protocols can use the STREAMS facilities and can interface with the system through TPI. Implementors of data link layer protocols can add their products to Solaris through DLPI. The latter products will enable Solaris to use new public services and new physical media for host-to-host communication.

Solaris supports the UNIX SVR4 standards for interfaces between device drivers and the system kernel. Device Driver Interface (DDI) and Device Kernel Interface (DKI) provide longevity and portability to conforming drivers.

Concurrent Initiatives

The Solaris Federated Services framework, which allows the integrated and manageable support of a variety of existing and future distributed services, is an important dimension in evolving Solaris to support the vision of global, mobile, and simple computing. This section presents an overview of concurrent initiatives underway by SunSoft and its partners in support of this vision.

Additional Global, Mobile, and Simple Computing Capabilities

Global Access to Services: Solaris will be enhanced to remove geographical barriers to collaborative work. This includes allowing users to consistently share files and applications, including supporting a shared workspace. Expanded support of network connectivity with Solaris and making network services available on a wide variety of networking technologies, including Integrated Services Digital Network (ISDN) and voice-grade lines, will simplify global computing. In addition, Solaris will provide very simple identification of and access to Solaris and non-Solaris resources across the global network.

Mobile Users, Systems, and Environments: To allow high-performance mobile computing, Solaris will include features that allow users to treat workstations like portables, in that they can be attached anywhere on the global network, and user resources (such as e-mail and files) continue to be accessible. In addition, non-continuous connectivity, allowing users to work when disconnected and to easily resume work after reconnection, is also a major development focus.

Simple Distributed Computing: Solaris plans allow for mobile, global computing with minimum cost-of-ownership. Solaris will reduce cost-of-ownership in two ways: through new system administration tools that are easy to use, and through advances in automation of services. Solaris will provide an object-oriented framework for advanced system administration. The framework will support the development of advanced graphical user interfaces for administering network objects. These user interfaces, reflecting the object-orientation of the framework, will be natural, intuitive, and consistent. Enhancements to support the goal of increased automation include self-administering and highly available network services.

Throughout the Solaris Federated Services framework are provisions for controlling the complexity that is inherent in global, heterogeneous computing. For example, as mentioned earlier, consistent and simple access to heterogeneous resources is central to the federated naming service. Simplification of Solaris core distributed services is also underway. A unified naming scheme that applies to files and other system and network resources is being developed.

Distributed Services Options for Solaris

The Solaris Federated Services architecture allows the support of multivendor distributed services within Solaris in an integrated and manageable fashion. This section presents an overview of ONC+, the core set of distributed services within Solaris, and existing and emerging standard distributed services products available from SunSoft Independent Software Vendors (ISVs).

ONC+: Core Distributed Services

ONC+ is a family of networking protocols and distributed services which provide the foundation for distributed computing within Solaris. It is an evolution of ONC and is available for the first time in Solaris 2.0. ONC+ is fully compatible with ONC and includes new capabilities for security, administration, and distributed application development.

ONC specifications are widely published and the source code is licensable from SunSoft. Currently, more than 100 implementations of ONC are available for such operating environments as DOS, Microsoft Windows, Novell NetWare, OS/2, Apple Macintosh, UNIX System V Release 4, Ultrix, AIX, HP/UX, MPE, VMS, MVS and VM.

Major ONC+ Components in Solaris 2.0

Network File System (NFS): This industry-standard distributed file system provides transparent access to remote files and directories across the network. Because the NFS protocols have been widely adopted in the industry, users can access files on PCs, workstations, minicomputers, mainframes, and supercomputers, independent of the underlying operating systems being run. NFS in ONC+ includes Kerberos authentication for additional security and multithreading for better performance.

Network Information Service Plus (NIS+): NIS+ is the new name service in ONC+, replacing NIS. NIS+ is fully compatible with NIS. It is a hierarchical enterprise naming service that provides the foundation for resource management in a dynamic, networked environment. It provides a secure extensible information base for storing and accessing network-based information such as hostnames, network addresses, and user names.

Transport Independent Remote Procedure Call (TI-RPC): TI-RPC provides a logical client-server communications structure that handles interprocess communication details transparently. It is a general-purpose protocol for the development of distributed applications that run in heterogeneous networks.

eXternal Data Representation (XDR): XDR is an architecture-independent method of representing data, resolving differences in data byte ordering, data type size, etc. Applications that use XDR are able to exchange data among heterogeneous systems.

Distributed Services from SunSoft ISVs and Partners

SunSoft ISVs have chosen to take advantage of flexible and modular architecture of the Solaris operating environment and its leading volume position in the RISC/UNIX marketplace to offer a host of existing and emerging standard distributed services products. Among these, SunConnect, Inc. offers an OSI services product for Solaris, as well as products allowing Solaris to support communication using de facto standard networking protocols such as DECnet and SNA. SunSelect, Inc. recently released a product called NetWare SunLink for Solaris, which allows Solaris-based systems to interoperate with NetWare systems.

Because of its leading volume and installed base position in the RISC/UNIX marketplace and its Federated Services framework, Solaris continues to be the platform of choice for support of emerging distributed services products such as the OSF Distributed Computing Environment (DCE) technology. Solaris is one of the first operating environments to have DCE V1.0 Developer's Kits available for customer use as a number of companies, including TransArc, Inc., have announced availability of such products for Solaris. TransArc, Inc. has also announced plans for a fully functional DCE product for Solaris 2.*x*.

For more information on the above products, please contact the software vendors directly.

The State(lessness) of Being: An NFS Update[1]

By – Hal Stern, SMCC, Northeast Area

Introduction

Sun's Network File System (NFS) was introduced more than seven years ago as a way of distributing filesystems access across a network. Since its technical unveiling at a USENIX conference, NFS has been licensed by nearly 300 vendors and universities, and connects machines as varied as a PC running DOS, an IBM mainframe running MVS, VAX/VMS hosts, and a bevy of UNIX boxes. Nearly all major UNIX vendors ship NFS client and server implementations as part of their standard operating system offerings. There are over one million machines running NFS, which means that there are at least as many non-Sun hosts using NFS as there are Sun machines — a fairly strong statement about the wide acceptance of the NFS protocol.

"Widely used," of course, does not necessarily imply other accolades such as "best performing" or "leading edge." In the time that NFS has been available, the actual RPC protocol it uses hasn't changed. The *implementation* of the protocol has been improved upon by features such as a server request cache, dynamic retransmission of client requests, and NFS accelerator products like the Prestoserve board and the NC400 network coprocessor.

1. Originally published in abridged form as "NFS – The Seven Year Itch" in *SunWorld*, November 1992, pp. 120-123. Abridged article is copyrighted © 1992 by Integrated Media, Inc. Reprinted with permission from *SunWorld* magazine.

On the other hand, research in distributed filesystems and potential improvements to the NFS protocol abounds: the Andrew File System (AFS), stemming from work at Carnegie-Mellon University; the Sprite project at UC-Berkeley; NFS caching analyses at Princeton University; and the implementation of NFS over TCP in the 4.3 BSD Reno distribution. AFS has been commercialized by TransArc, and also serves as part of the technology base for the Open Software Foundation's distributed filesystem component of their Distributed Computing Environment. Now that NFS has survived longer than most marriages, it's worth taking a look at how some trends in distributed filesystems might improve NFS, and what to expect in future NFS implementations.

This article explores what these research and development efforts imply about NFS in the following areas:

Performance	How can NFS be made faster?
Naming & Administration	How is filesystem data located? What system administration input is required?
Consistency	How can NFS provide reliable multiple client access?
Scalability	How big can an NFS installation be?
Availability	How do dual-ported disks and failover systems work with NFS?
Security	How can we make networked resources less prone to attack?

Performance

We'll start with the thing usually taken as the source of the ubiquitous "`NFS server not responding, still trying`" message: *performance*.

Probably the largest single complaint about NFS is that it is a poor performer. NFS read and write rates of 500 kbytes/second over Ethernet are considered fairly good results, but those throughput results are about half of what the best TCP/IP labs can produce with other application. Bursts of traffic from NFS clients increase network contention, reducing actual throughput on networks with many machines. But making a bigger NFS server with more and faster Ethernet interfaces is not the complete answer: network loading may be a good part of your problem, and the network itself imposes a speed limit on traffic.

The speed differential between a local disk and one mounted over the network has increased greatly since the introduction of NFS. Five years ago, desktop SCSI disks had far worse performance than their SMD brethren on file servers.

Today, however, fast SCSI disks run at 10 Mbytes/second on desktops and servers, and the transfer rate advantages offered by IPI and SMD have all but disappeared. During that time, one transfer rate has remained constant: Ethernet. It's time for NFS to figure out how to better use local resources to avoid over-the-wire data transfers between clients and their servers.

One simple change is to allow NFS to make use of pages cached in the client's memory with less overhead. Normally, any page read from a file is considered valid for 30 seconds, and then the file's attributes are retrieved from the server and compared to the timestamp on the cached data. If the server's copy of the file hasn't changed, the lifetime of the cached data is extended. If the server's copy has been updated, though, the client retrieves the fresh data. Doing the cache consistency checks can be part of a performance problem, especially with large-memory configuration clients or clients mounting filesystems over internetwork links. Check your typical NFS server and you'll find that at least one third of the requests it handles are `getattr` requests, many of which are part of cache consistency checks.

Cut down on the number of cache consistency checks by increasing the "valid window" to several minutes, using the `actimeo` mount option. It would be nice if NFS offered an infinite attribute cache period for clients mounting filesystems that are static, like CD-ROMs or read-only tools depositories. Changes in attribute caching and cache consistency checks will decrease NFS traffic at the expense of increasing the severity of data consistency problems which you may or may not know exist. We'll visit the issues surrounding cache consistency below.

It would also be nice for NFS to make more efficient use of non-Ethernet media with larger bandwidths. The 8-kbyte NFS buffer limits performance over fast networks with large transmission units, such as FDDI. However, this calls for a modification to the NFS protocol so that it uses a larger buffer size, and protocol modifications will take time to be adopted by all of the NFS licensees.

The obvious way to improve NFS performance is to simply send fewer NFS requests, and satisfy as many of them as possible using resources local to the NFS client. Once the client's memory cache fills up with NFS file pages, some of these cached pages must be discarded and re-read from the server the next time they are needed. To improve NFS performance, the local memory cache can be extended using a local disk cache as "backing store" for file pages that don't fit into the client's memory. Local disk caching is a big drawing card of the Andrew File System (AFS), which copies 64 kbyte "chunks" of remote files to local disk when they are first referenced by the client.

NFS clients with local disk caches can satisfy some NFS read requests from local disk, and avoid sending requests to the server. This reduces the load on the network and server. Sequential I/O proceeds at local disk transfer rates, rather than at Ethernet speeds: that's at least a 5:1 improvement. But some hard technical questions need answers: How will the cache be managed? What does it do for random I/O? How robust are the cache consistency, crash recovery, and server update algorithms?

The cache has to be managed fairly intelligently, or the overhead of copying to and from the cache may exceed the performance boost gained by not going over the network for all NFS requests. Blindly copying all files referenced by the client into its cache may result in many unnecessary transfers — over that relatively slow Ethernet. Data chunk management and consistency policies loom as determinants of the actual usefulness of a client-side cache.

Random I/O may be improved if more operations can be handled out of the cache. To an NFS server, interleaved requests from multiple clients look like a large random I/O stream, so the NFS server workload is usually very heavy in random I/O operations. When talking only to its local cache, though, a client's own traffic pattern looks much more sequential for many kinds of operations. Consider six NFS clients each opening different files. The NFS server sees apparently random accesses to six different parts of the disk, but each *stream* of requests may be for completely sequential I/O on each client. Operations on large files may still behave randomly, and local caching won't do much to help them. But for many typical file access patterns, a local disk cache should reduce network and server loading, and improve local system performance by using fast local disks instead of fast remote disks separated by a slow and possibly busy network.

The Sprite network at University of California at Berkeley, and NFS trace-driven studies at Princeton University, have demonstrated that a well-managed local cache can significantly reduce server traffic and increase distributed filesystem performance. SunSoft has announced plans to implement client-side disk caching in a future release of Solaris 2.*x*, eliminating one of the major differentiators between AFS and NFS. But closely tied to the issue of performance is that of naming, particularly when comparing AFS and NFS.

Naming and Administration

You can't use a fast fileserver if you can't even locate the resources it exports to the network. A simple, extensible naming scheme reduces the burden of managing a large distributed environment: the system administrator manages the naming scheme, rather than machine-specific configuration information. If this sounds an awful lot like NIS (Yellow Pages), it is. NFS provides the distributed data management portion, and NIS provides the distributed naming service. In Solaris 2.0, the naming system uses a per-host switch that references NIS, NIS+ (an improved version of NIS), the Domain Name Service (DNS), or local configuration files. We'll look at some naming issues and the tools to manage the filesystems and servers that are being named.

The heart of any NFS naming system is turning an abstract definition of a resource into a description usable by the network. For example, convert the pathname `/home/stern` into the NFS mount specification `bigbear:/files1/stern` . A static naming system does this conversion once, when the resource is first used, and that association is used until the resource is explicitly discarded. A good example of a static naming scheme is the NIS and automounter combination: the automounter maps convert machine-independent references like `/home/bob` and `/tools/wpd` into ***server*:*filesystem*** mount specifications, and the NIS maps provide the map and server information to all clients. The automounter can choose from any one of a set of replicated servers, but once it has completed the NFS mount, it cannot choose another server if the first choice becomes unavailable.

The automounter provides a static naming scheme because it relies on NFS to complete the mount, and NFS doesn't allow clients to locate new servers on the fly. The static name translations implemented by the automounter are valid as long the filesystem is mounted by the client — as soon as the automounter can unmount an idle filesystem, a new name translation can be used. If you choose to move `/home/bob` to another server while that directory is in use on some client, NFS is going to have problems locating the new directory structure. Any machine that references `/home/bob` after the move will pick up the latest mapping from an updated automounter map, but those machines that already had `/home/bob` mounted at the time of the change will be confused: their old location information for `/home/bob` points to a non-existent directory on the old server.

Dynamic naming allows you to change the location of a resource and still ensure that clients using that resource will find it "on the fly." In the example above, the dynamic naming system would recognize that the location of `/home/bob` changed, and route requests to the new server — in effect, the naming system changed the mount information while it was in use. Dynamic naming is the heart of the AFS/DCE cell naming scheme, which allows you to migrate filesystem objects through the network while they are in use.

Is dynamic naming a huge benefit? Maybe. It's possible a more flexible naming scheme hides some underlying system administration problems. Accommodating migration of users' home directories sometimes masks the need for a serious accounting of server usage and implementation of load balancing. If your server disks are continually filling up, it might be due to users loading software distributions or leaving core files behind in their home directories. Dynamic naming isn't going to fix that problem. Instead, you may find yourself dealing with a large frictional workload of directory shuffles while side-stepping the basic system administration problem. In addition, there are costs associated with the advantages offered by a dynamic naming scheme. More complex systems have more complex — and often more puzzling — failure modes. As commercial-strength implementations of cell naming are adopted by the major vendors, the usefulness and issues surrounding dynamic naming should become more clear.

But what about those system administration issues? Dynamic or static naming aside, you still need to make decisions about laying out users and tools on filesystems, determine where and how to grow your fileserver installation, locate bottlenecks and hot spots in the network, and answer pleas for improved performance. Capacity planning, network and server evaluation, and performance tuning have been something of a black art, and NFS performance with large numbers of clients has only recently been characterized.

A plethora of network and NFS traffic monitoring tools exist, including the `nfswatch` utility. Vendors such as Tivoli, working with Sun and others, will undoubtedly produce new and more elegant ways of analyzing large networks. The problem, however, is implementing policies based on the data presented by the analysis tools. Here's an example: you decide that you need to break your largest NFS network into two or more subnetworks. You need to prepare a budget for new network hardware, and a timeline for completing the changes. What do you look at? What tools will help with the process?

Here are some basic questions to ask: How much traffic is on the network? Are there any groups of clients and servers that do little with other machines? If you use subnetworks, how will they be attached to the other networks in the company? If you use a bridge, what machines should go on each leg? Few system administration tools can keep you from putting a slow network device in between all of your servers and their clients, but they can give you the data to realize that such a configuration would be a poor performer.

As a system administrator, you need to know how your users and their client machines use your resources. Do their needs or usage patterns change over time? How will you accommodate a major shift in demands, such as a the start of a new development project or the beginning of the academic year? As always, communication with the users is a top priority, and is your most powerful tool for setting expectations and collecting input.

Consistency

In addition to knowing what kind of load and demands users place on the NFS environment, it's also wise to know what kind of behavior they expect from the filesystem. NFS does not mimic the behavior of a local UNIX filesystem precisely. UNIX filesystem features, such as symbolic links, may not make sense or be available on non-UNIX NFS servers. The virtual filesystem framework on which NFS is built defines a set of UNIX filesystem-like operations, but there is no requirement that an NFS implementation support all of them.

Think about what happens if one or more processes have a file open, and some other process removes that file. On a local filesystem, the data blocks for the file are kept intact but the directory entry for the file is removed — the file becomes invisible. When the last process using the file closes it, the data blocks are freed. Since NFS has no concept of state, and therefore no notion of open or close operations, it relies on the clients to enforce some of this UNIX filesystem behavior. If the multiple processes are on the same client, then the client takes a side-step when a process tries to remove an open file: the remove operation is converted into a rename, so the server keeps the file around, albeit with a "hidden" name starting with "`.nfs`". When the last process using the remote file closes it, the NFS client finally issues the remove.

More subtle than the open file example is the general behavior of multiple processes reading and writing the same file on different NFS clients. Consider the following sequence of events: machines *A* and *B* each open file `foo` on server *C*. *A* opens the file read-only, but *B* opens it read-write. *A* reads most of the file and caches it in memory, but *B* reads the file and modifies the first 100 bytes. *A* will not see these changes until it does the next cache consistency check on the cached page, which could be up to 60 seconds later. During this window, *A* and *B* have inconsistent parts of the file. *A*'s data is even inconsistent with the server, which is the "truth" regarding file contents.

"So use file locking," you say; "that's the only way to ensure that multiple readers and writers can agree on the time sequencing of changes to a file." NFS, however, has no native file locking support. Locking would generate state to be preserved on the NFS server, and NFS has a stateless server. Sun introduced the Network Lock Manager (NLM) protocol, in the `rpc.lockd` daemon, which sits adjunct to the NFS protocol. NLM is a completely independent protocol that handles state changes and recovery after a client or server crash. Unfortunately, the NLM protocol is weak and somewhat buggy: `rpc.lockd` has endured eight patches on top of the SunOS releases since 4.0.3.

The UNIX filesystem makes sure that you see the right file contents after a write, but it doesn't enforce relative time ordering of reads and writes to the same file. If you want to absolutely ensure that no processes read a file while it's being updated, you need to at least use advisory file locking. Solving the multiple reader/writer problem with any degree of certainty requires a complex, distributed locking protocol. The VAX/VMS lock protocol used in VAXClusters is frequently held up as a model for a solid distributed lock system. Other remote locking systems have been implemented to handle the two-phase commit problem in distributed databases. Even if a locking facility is available, it's still up to the local gurus to make sure that users understand and use it. When NFS adopts a more robust locking model, the consistency issue may vanish, provided users abide by the file locking rules.

Scalability

Rate of growth is one of the best measures of success for a distributed system. With more users, client machines, and fileserver resources on-line, a well-bred NFS installation will handle the increased load and show little decrease in performance. While that may seem a bit optimistic, it's reasonable to explore what can be done to NFS to make it more scalable from small to large installations.

On a local Ethernet, your performance is likely to be colored as much by network traffic as by server load. If clients are having trouble getting packets onto and off of the network, or if a bridge or router along the way is dropping parts of long NFS packet trains, then your perceived NFS performance will suffer. As you add clients to an Ethernet, the effects of collisions and deferred transmissions (avoided collisions) start to become apparent. So how many NFS clients can you put on a network? As with most performance questions, this simple query begs other questions: What are the clients doing? Are they diskless or dataless? How much memory do they have?

Diskless clients are less pleasant network neighbors, since they are frequently paging over the network, while dataless clients issue many fewer write requests. When comparing distributed filesystems, make sure you compare similar configurations of client machines: an AFS client using a local disk for client-side caching should be compared to a dataless NFS client that pages and swaps locally and gets some of its executables from the local disk. You may only fit 20 diskless clients on a network but be able to put 50 or 100 dataless machines on a single wire. The issues surrounding NFS performance also drive its scalability: you can put more NFS clients on a single network if you configure them such that they issue fewer over-the-wire NFS requests. Larger client memories (for large memory page caches), local disk caching, and dataless configurations are the independent variables in the scalability equation.

Scalability applies to distance as well as sheer size. To be a proper corporate citizen, NFS should run over internetworks as well as local area Ethernets or token rings. If NFS is the preferred method for sharing data on the local area network, then it should suffice as a sharing mechanism over the company's wide-area network as well. You can tune the NFS mount parameters to make it more tolerant of slow internetwork links,[2] but the UDP transport protocol used by NFS is a problem in the face of busy or unreliable connections.

UDP is best suited for fast transmission over a relatively reliable network. If the network swallows or damages part of a UDP transmission, the application sends the whole chunk again. In the case of NFS, this means re-sending an entire NFS buffer of up to 8 kbytes if even one byte is damaged along the way. TCP is far better at handling internetwork communications; sliding transmission windows in TCP allow it to meter out data in the face of a lossy line or otherwise congested link.

TCP has the most to offer NFS in the area of wide-area networking, although NFS-over-TCP is often cited as a possible performance improvement. Using the TCP protocol doesn't make NFS any faster, since most of the throughput constraints are in the NFS protocol itself and in the physical media being used. Over a wide-area network consisting of some 56 kbaud or T1 (1.54 Mbit/sec) lines, NFS-over-TCP will probably outperform the NFS hosted over UDP, simply because TCP is better behaved over the slower line. On Ethernet, however, the additional CPU overhead of driving the TCP protocol on the server may actually reduce the maximum load a server can handle, since many NFS servers run out of CPU bandwidth first.

If TCP handles the slower bandwidth lines well, how can NFS scale to handle faster networks such as FDDI, and larger data items such as ECAD or MCAD files that may run 20 Mbytes each? First of all, the NFS buffer size can be

2. Hal Stern, "/sys.admin: Internetworked NFS," *SunWorld*, July 1992, pp. 95-100.

increased, so that NFS sends more data with a single RPC call. Changing the NFS buffer size from the current 8 kbytes to 64 kbytes, for example, would match the buffer size to SCSI disk track buffer sizes, possibly making more efficient use of the server's disk drivers. Matching the buffer size to the maximum transmission unit (MTU) of the media is also a possible win, since NFS could "stream" data more efficiently over a faster network with a larger frame size. On an Ethernet, an 8-kbyte buffer turns into six IP packets; but on FDDI, an 8-kbyte NFS read or write request sends only two IP packets. As the transmission units of networks increase, NFS should be able to handle the MTU for the network it uses. Changing the buffer size above 8 kbytes, however, requires a change to the NFS protocol, which will be slow to spread throughout the NFS community.

Another possibility is to encourage NFS to use a streaming I/O model, where client requests are issued almost continuously, rather than in bursts managed by the `biod` daemons. The in-progress request list could grow quite long if a large file was being written, instead of being limited to the number of `biod` daemons that handle write-behind. When moving large files around the network, a machine doing streaming I/O looks more like a host using TCP, sending data until it realizes the other end can't keep up. The client-side implementation of NFS in Solaris 2.0 uses kernel threads instead of `biod` daemons to manage the client requests, and up to eight threads per mount point may exist at any time. This is a good step, and some practical experience will determine if the existence of more threads (and consequently a deeper write queue) on the client side actually produces any improvements for large file operations.

The ability to send data at varying rates, depending upon file size and network loading, will determine how well NFS performs over floating bandwidth lines, such as ISDN. With an integrated voice and data link, it's possible that some of your ISDN data bandwidth is taken up by voice communication, so other activity is taking away from the potential NFS throughput. Of course, the opposite is true, so copying a large file from an NFS server at work to your home machine is a sure-fire way to commandeer the network bandwidth needed for a telephone call from your in-laws.

At this point, we've covered ways to get NFS data onto the network, locate it once it's been made available, ensure consistency among multiple clients, and scale the environment using new and faster technology. Now we'll turn our attention to the administrative issues of data availability and security: keeping the data on the network and keeping it safe from harm.

Availability

Data availability is a more involved notion of fault tolerance that includes hardware as well as software problems. In a fault-tolerant system, replicated components like CPUs, disk controllers, and I/O busses execute the same task in parallel, with failures detected by a lack of agreement among the replicated parts. If three disks don't return the same data, then one of them has most likely suffered a failure. Data availability goes beyond keeping the hardware reliable and accounts for system crashes caused by operating system problems, application bugs, and other software faults that cannot be detected by redundant hardware. Keeping data available means that *some* machine gets the data to and from the network, but that machine might change identity in the event of some kind of failure. NFS availability can be viewed from either the client's or server's perspective: we'll look at the different features offered by server and client failover mechanisms.

Server failover protects you against faults on the NFS server itself. It uses redundant servers with dual-ported disks and some heartbeat software to make sure that one machine is always exporting the data on the disk for NFS clients. If the primary server crashes, or stops providing heartbeats for any reason, then the secondary server takes over the disks and assumes the network identity of the primary server. Both primary and secondary servers use the same IP addresses, requiring that the secondary server remain quiet on the network until it takes over for a failed primary server. NFS clients don't notice any changes. The failed server had no information about its clients, so the backup server can take over by simply coming up on the network and accepting requests previously directed at the primary server. NFS requests that were in progress at the time of the failure are retransmitted by the clients, and executed by the secondary server.

Mirrored disk images — redundant disks used to replicate data on a single machine — make a single NFS server more resilient, but do not provide full server availability. Disk mirroring allows an NFS server to survive a failure in a disk or disk controller, but you are also susceptible to failures in network interfaces, CPU, memory, or the server's operating system. Using redundant, identical servers allows the data to be made available via NFS if any one of those components fails. Solutions exist for both disk mirroring and server-side failover today: Sun's Online DiskSuite product includes a mirroring feature, and Fusion Systems Group and Sun's consulting group have an NFS high availability product for Sun servers that supports both SCSI and IPI disks.

But server failover is only part of the problem. What if the segment of your network attached to some servers has a meltdown, and those servers are unreachable? What if you're using replicated NFS servers, and after a power failure, 90% of your NFS clients end up binding to the same "tools" server? Many users feel cheated by the automounter: they think that if the automounter can choose from any one of a set of replicated servers, then it should be able to pick another server if the primary choice becomes unavailable. This desired feature is really client-side failover, which uses the replicated server selection process as just part of its data availability solution.

Client-side failover is most useful when you have replicated servers with read-only data. If one server crashes, you'd like to select another one and continue sending NFS requests, with the switch transparent to the processes running on the client. Should be easy, right? Well, not really. NFS clients use file handles to locate files on the server; file handles are little more than encodings of the file's filesystem and inode number *on the server*. Switch to another server and those inode and filesystem numbers are meaningless — so all of your file handles are invalid, or stale. Client-side NFS failover requires implementing a mechanism for regenerating new file handles valid on the new server, which means that NFS clients need to keep track of how they located those files in the first place. While this feat isn't Turing Award material, it's not trivial either, and to be production-quality, the file handle regeneration must be able to survive any combination of server and network outages.

Client- and server-side failover each have their benefits. Server-side failover doesn't require any changes to the implementation of NFS, and it allows you to provide availability for writeable data like home directories. Server-side failover doesn't help you, however, if your network flames out or if you want clients to dynamically load balance among many replicated NFS servers. Implementing an effective client-side failover policy will force you to think about network and server performance in general, since clients will have a hard time distinguishing a very busy server from a dead one. You'd like to be able to roll out client-side failover without creating a rash of perceived failures and client rebindings. Determining the number of replicated servers needed and the necessary conditions for executing the client-side failure code will become the new and interesting NFS performance problems. Defining an effective policy means managing all of your replicated servers on a broader scale, without having to micromanage resources and dole out servers to a defined, small pool of clients. With some of the issues of data availability taken care of, you can focus your attentions on the problems of data integrity.

Security

Putting any resource on a network opens it up to a variety of possible attacks. Users can masquerade as other users, submitting forged NFS requests to a server. Unauthorized machines may mount filesystems and circumvent access controls intended to protect data. Making NFS secure requires improving the authentication of NFS requests — assuring an NFS server that the client has the proper privileges to operate on the server's files.

Sun has offered Secure NFS for at least two generations of SunOS 4.*x*. Secure NFS is built on Secure RPC, which uses DES encryption and exponential key exchange to verify each NFS RPC request. Secure NFS prevents users from hand-crafting NFS requests by requiring that each request contain a verifier, which can only be generated using the real user's password. DES encryption of the RPC requests can cause a 20% performance hit on each RPC, although use of the DES encryption chip in SPARCstations can reduce the encryption cost to about 5%.

Security in a network system is an inherited property: if you secure it at the lowest layer, there's no need to worry about attacking the problem at the RPC or NFS layer. If you can guarantee that your IP network is completely trustworthy and reliable, and that every packet on the network was generated by genuine users with good intentions, then it's not necessary to bother with specifically securing the RPC layer or NFS. In some environments, this implicit trust is inherent in the organization's lack of connectivity to the outside world. If there is no IP network extending beyond the company's walls, and everyone working on the network is trusted to act within the company's guidelines for responsible behavior, there are few sources of security threats.

But many computing systems are not so privileged, and system administrators cannot trust machines, users, or packets on the network. A clever user with a PC, able to generate hand-crafted packets mimicking valid RPC and NFS requests, is seen as a threat to the network and its data. The Kerberos security system, developed at MIT as part of Project Athena, was predicated on the belief that nothing on the network could be implicitly trusted. Kerberos secures an entire system using a "ticketing" scheme. Users are required to validate themselves before they are granted a ticket to use the service under Kerberos control. In addition to Secure NFS, Solaris 2.0 includes client-side support for Kerberos, so that NFS requests may be validating with a Kerberos server. Solaris 2.0 does *not* include a Kerberos ticket server implementation.

So you're charged with protecting your company's confidential data, but you trust your peers not to hurt themselves with sharp networking instruments: what are the alternatives and requirements? A few simple precautions go a long way, and point to some areas for NFS security improvements:

- Control the export of filesystems. Don't grant access to any machine on any network. Enforcing selective access to "just your machines" is a tedious process, however, with cumbersome netgroup definitions required. Each change in the network population also requires a change to the NFS export controls.
- Watch out for self-inflicted damage. Don't export the root filesystem on the server, because everything on the server will be accessible to NFS clients. Similarly, don't run NFS on firewall hosts: while you intend to export the filesystems to the "inside" of the security gateway, the filesystems will be accessible on all network interfaces of the router.
- Limit access to your network. Remember that if you trust all of those IP packets, you can probably trust what they're doing — NFS or RPC requests — as well. If there's some part of the network that you don't trust, segregate with a router, gateway, or firewall. Obviously, this doesn't apply to universities or large campuses with many access points to the network: those are the situations for which heavy-duty Secure RPC and Kerberos authentication are ideal.

There are several excellent books on computer system security. If you feel your environment is subject to attacks, be sure to secure it at every possible entry point. Use the reference books as a model for your environment, and work with the users to explain the security policies, procedures, and underlying intentions.

What Next?

NFS is certain to evolve over the next half-decade of its life. SunSoft has announced plans to include local disk caching and an improved automounter in future Solaris releases, and Solaris 2.0 brings Kerberos support and NIS+ naming in a more immediate timeframe. Improvements to NFS protocol implementations and new adjunct protocols are almost assured, but the viability of a revised NFS protocol depends upon how quickly the NFS licensees adopt it and implement it. Widespread acceptance breeds a reactionary attitude toward new protocols: why risk the heterogeneous interoperability that required significant effort to create? What remains clear is that distributed filesystem research proceeds at a reasonable pace — the path has been cleared for a new and improved NFS.

Network Load: X-terminals vs. Workstations

By – Gary Oing, Brian Wong — SMCC Technical Product Marketing

In a recent paper, a large government customer studied the network utilization consumed by diskless, dataless, and datafull workstations, as well as X-terminals.[3] The conclusions drawn are of interest to anyone who must configure large network configurations.

The experimenters used a 16 Megabyte SPARCstation 1+ in the workstation configurations, and a Tektronix 4210 X-terminal. A Network General Sniffer was used to monitor the network without affecting throughput. Several applications were used, two of which were internally developed by the customer's staff. These were a GIS/Mapping package and what appears to be a flat-file database package. The commercial package was BBN Slate. The test network was isolated in order to avoid interference from unrelated packets. The NFS server also served as the application server for the X-terminal; the system appears to be another SPARCstation 1+, but this is not spelled out in the paper.

Both startup cost and steady-state load were measured for each configuration. The results were very clear. In all configurations, the workstations generated significantly more network activity during application startup than X-terminals. In fact, in terms of the number of bytes transferred, the workstations consumed 7-10 times as much network bandwidth. However, during steady-state operation, the X-terminal consumed 2-7 times as much network bandwidth. The proportion of actual time spent in these two phases varies by application and by user base, but in almost all circumstances the vast majority of the time is spent in steady-state operation. Certainly it is true that application execution time is usually greater than 3.5 times length of startup (7-9 seconds for the specific applications tested).

The data reported also indicated that the applications took considerably longer to run on an X-terminal than on the workstations — sometimes by a factor of 3 or 4. Certainly the additional networking load imposed contributed to the delays, but a more likely explanation is that the particular X-terminal used for

3. James E. Manley, Steven P. Trieber, *Ethernet Bandwidth Utilitzation of Datafull, Dataless, and Diskless Node Workstations and X-Terminals*, The MITRE Corporation, Bedford, MA, November 1991.

the testing is not a very fast one. By comparison, the SPARCstations had a GX accelerator. Faster X-terminals (or faster workstations) could likely improve upon this performance.

The experimenters conclude that X-terminals should be configured carefully, and in limited numbers, based upon the considerable excess of network consumption. They also note that diskless workstations consistently consumed about 20% more bandwidth than the otherwise equivalent dataless configurations and recommend against them for these reasons.

The customer study provides much food for thought. SMCC Technical Product Marketing will be investigating this area more thoroughly in the near future.

SPARC 4

SRMMU, TLBs, and Contexts

By – Jim Mauro — System Engineer, Somerset, NJ

This article is a brief overview of the low-level components involved in translating a virtual address to a physical address on a SPARC system. Numerous white papers exist that provide excellent descriptions of the higher level Virtual Memory (VM) architecture. However, there seems to be a lack of a single point of reference that discusses the low-level architecture and implementation of virtual memory. This discussion fills that gap, focusing on the SPARC Reference Memory Management Unit (SRMMU) architecture, rather than the older Sun/SPARC MMU.

SPARC Architecture

The SPARC Architecture is drafted by SPARC International, Inc. Currently, SPARC Architecture is at Version 8, although the V9 draft has been completed and approved.[1]

1. For a discussion of SPARC Version 9, see Joel Boney, "From the drawing board," *SunWorld*, October 1992, pp. 100-105.

The SPARC Architecture defines things like the supported data formats, internal registers, instruction set, etc. Also defined within SPARC Architecture V8 is the SPARC Reference MMU. This piece of the architecture, along with all the others, provides guidelines for folks who wish to build SPARC processors.

The *only* SPARC processor today that is V8-based is the SuperSPARC[2] (used in Models 30, 41, 42, 52, and 54 of SPARCsystems 6*x*0MP and SPARCsystem 10). All others are V7-based, even the SPARC processor[3] used in Models 120 and 140, although they implement the SRMMU, which is from V8.

The SPARC Architecture V7 does *not* define a Memory Management Unit, or in any way provide guidelines for an MMU implementation.

Architecture: Implementation of Hardware and Software

It's one thing to provide the guidelines (the architecture specification), but quite another to adhere to it within an implementation.

And then, of course, there's the hardware implementation and the software implementation. For example, the SRMMU specifies a 32-bit Context Register, thus allowing for 4 billion contexts. The Sun-4m System Architecture defines the Context Register as having 0 - ***n*** bits for the context number — some value less than 32 bits; thus the Sun-4m System Architecture doesn't specify an exact number of contexts, but rather leaves that to the implementors.

As far as implementation goes, the SPARC processor specifies 12 bits for a context number, providing a maximum of 4k unique context numbers. The SuperSPARC module, either with or without the 1 Mb external cache (`E$`), defines 16 bits for a context number, allowing up to 64k contexts. Let's take implementation one step further: even though the SuperSPARC hardware provides for 64k contexts, SunOS 4.1.3 restricts it to 4k.

So here we have an example of two different processors that are both based on the SRMMU as defined in SPARC Architecture V8, and are both Sun-4m (Mbus) implementations, but which vary slightly on the specific definition around the number of supported contexts. This is not really such a big deal — we're just trying to illustrate how one should be aware of the distinction between architecture and actual implementation.

2. Often commonly referred to as the "Viking" processor.
3. Often commonly referred to as the "Ross" processor.

The SPARC Reference MMU

The SRMMU provides for things like:

- 32-bit Virtual Address
- 36-bit Physical Address
- 4-kbyte page size
- a 4-level mapping hierarchy that deals well with sparse address spaces, as well as large linear mappings
- physically addressable SRMMU tables

The MMU itself is the hardware responsible for virtual-to-physical translations. Of course, there is a fine line between what the hardware does and what the OS does in terms of address translation. For example, the OS must provide the definitions for its implementations of things like the Context Table, the 3-level mapping tables, a Page Table Entry, etc. Also, the OS must actually deal with building and maintaining these tables. To say that the MMU hardware does the translations — while perhaps technically accurate in a strict sense — understates the OS work involved in the low-level address translation process.

The SRMMU provides for a 4-level mapping scheme, where the Level-1 and Level-2 tables contain pointers to the next level table, and the Level-3 table contains an actual Page Table Entry (PTE), which is a 4-byte structure that defines a mapped page.

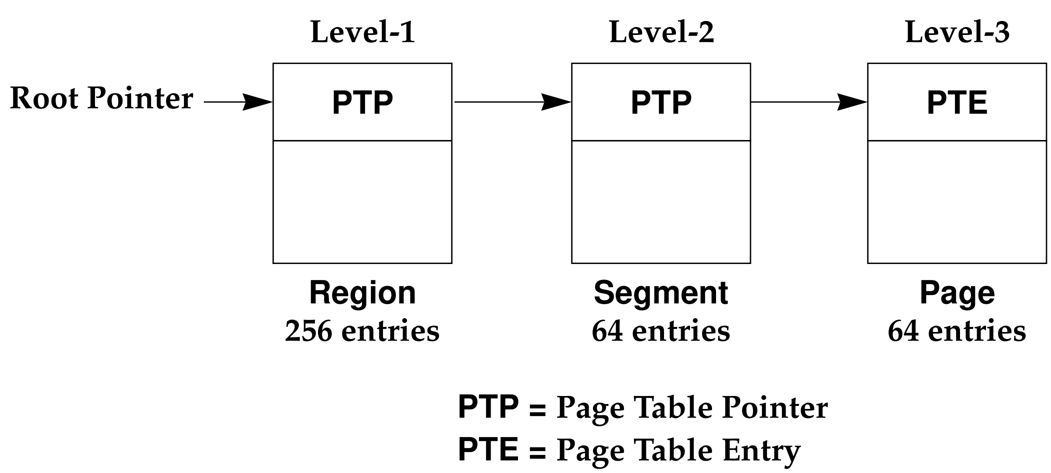

Figure 4-1 SRMMU Mapping Scheme

The fourth level in the hierarchy is actually the top level, where the root pointer lives (we will discuss this shortly). These three tables are referred to within the SRMMU as defining the *page* (Level-3), *segment* (Level-2), and *region* (Level-1) mappings for a process. These terms get used frequently in different contexts, and warrant some explanation.

A *page* is the smallest mappable unit that the virtual memory system can deal with. Processes are allocated and de-allocated memory in units of pages, and a page is the smallest unit that the system can impose access restrictions on. The size of a page may vary. Many older Sun systems implement an 8k page size. The SRMMU defines a 4k page size. Every mapped page in virtual memory has a PTE associated with it, which provides information about the page.

A *segment*, as per the SRMMU, is simply a group of pages. This is not to be confused with a Segment as defined by the Sun VM architecture (Sun VM ties a segment, with a specific segment driver, to a particular type of memory object, such as a vnode). Also, there is no correlation between a Segment as defined by the SRMMU and the traditional UNIX process text, data, and stack segments. The SRMMU goes no further in defining what a “segment” is, other than to imply that it is comprised of a logical grouping of pages.

A *region* is (as you may have guessed) a logical grouping of segments, again not to be confused with the traditional UNIX process regions (text, data, etc.), nor with SVR3’s notion of regions, as per SVR3’s implementation of shared memory. Like segments, the SRMMU is not specific about what a “region” is.

This scheme makes it relatively easy to deal with processes that have small, sparse address spaces. The actual page data is handled via a 64-entry Level-3 page table, each entry of which is capable of mapping up to 256 kbytes of virtual memory (64 entries x 4 kbytes-per-page). By mapping the pointers to these page tables one level back, the tables can be spread around more easily than the previous Sun MMU PMEG implementations. Even the pages themselves can be spread out, and do not need to occupy contiguous memory, as they did in the older Sun MMU.

The SRMMU tables are all accessed using physical addresses. Because of this, many SPARC implementations that use virtually addressed caches cannot cache the translation tables. For example, the SPARC processor in Models 120 and 140 uses an external virtually addressed cache. As a result, the processor routes all SRMMU-initiated accesses directly to the main memory. In order to insure that the kernel and SRMMU have a consistent view of things, kernel accesses to the tables are done via non-cacheable mappings.

On the SuperSPARC chip, physically addressed cache is implemented, allowing for the mapping of the SRMMU translation tables. On a SuperSPARC implementation with the external 1Mb cache (`E$`), the kernel configures the SRMMU to look in the cache for the tables, and the kernel references the tables via cached pages. If the `E$` is absent, the SRMMU does not look in the cache, and the kernel uses non-cacheable pages to store the translation tables.

Contexts

Context is also a term that has several meanings in the UNIX world. In a broad sense, a process's context defines the environment in which it runs. This may include the hardware registers (General Registers, Program Counter, Stack Pointer, etc.), address space mappings, open files, etc.

For the purposes of this discussion, we're going to deal with contexts as defined by the SRMMU, and as implemented by SPARC systems. The SRMMU defines a context as a unique number which identifies an address space. Since every process has an address space associated with it, each process under Solaris has a unique context number, or an invalid context number.

Every implementation of SPARC has some number of contexts it supports, be it 64, 256, 4k, or whatever. The number of supported contexts simply defines the number of active process address space mappings that can be maintained at any one time. Since it is possible that the system will have more processes than available contexts at any point in time, it becomes necessary for the system to begin reassigning contexts; thus it is possible for processes to have invalid context numbers at different points in time during their existence.

The SRMMU defines at Context Register and a Context Table Pointer (CTP). The CTP, as implemented, contains the physical address of the System Context Table. The System Context Table is simply a table of root pointers — that is, each entry in the System Context Table is a Root Pointer for a process's tables, and each pointer points to the Level-1 (region) table for that process.

The way we locate a given process's root pointer is via the Context Register. When a process is context-switched in to run, if it has a valid context number, that number is loaded into the Context Register and used as an index into the System Context Table to locate that process's root pointer.

```
(struct prop *)p->(struct as *)p_as->(struct hat)a_hat.hat_ctx->c_num
```

This rather lengthy and cryptic string is our “path” to a process’s context number. Each `proc` structure contains a pointer to an `as` (Address Space) structure. Within the `as` structure is a `hat` structure, which contains a pointer to a `ctx` structure. The `ctx` structure is comprised of two items: the context number (`c_num`, an unsigned integer), and a back pointer to the `as` structure.

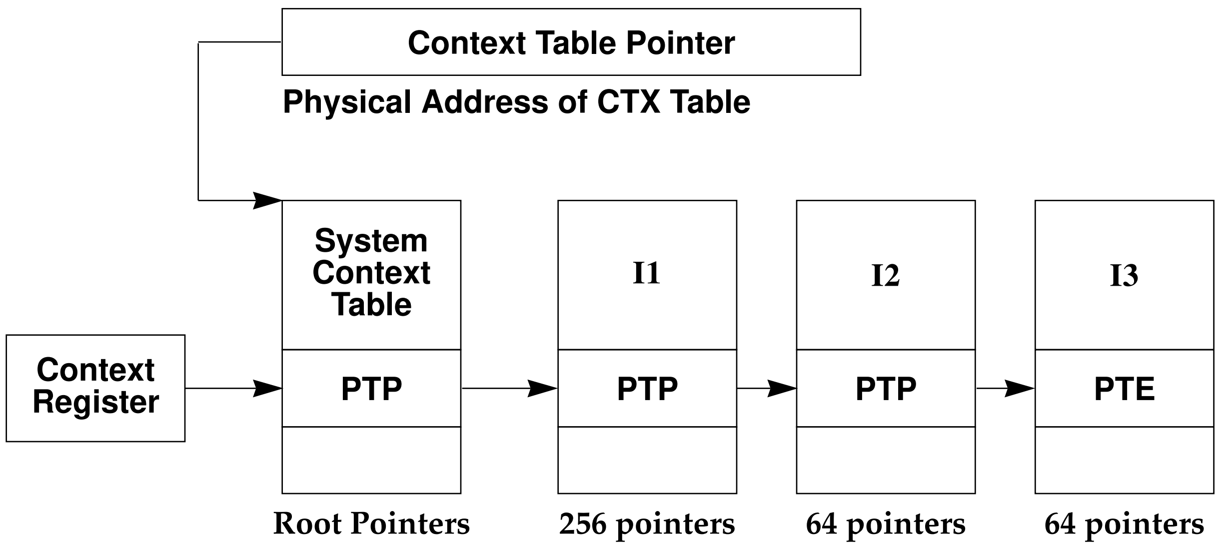

Figure 4-2 Process Context

Address Translation

When the running process generates a Virtual Address (VA), the bits within the VA are broken down and used as indexes into the three tables used to map the VA space of the process. As discussed in a previous section, the actual mappings are found using the Context Table Entry, located via the Context

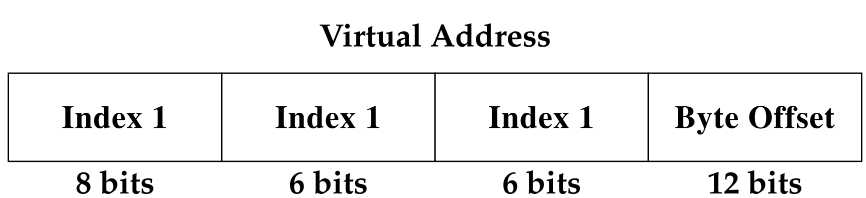

Figure 4-3 Virtual Address

Table Pointer and the Context Number. For a process, this points to the first (Level-1) table. The high-order 8 bits of the VA are used as an index into this table to locate the appropriate Page Table Pointer (PTP), which points to the next level table. The next 6 bits of the VA index into the Level-2 table, locating the PTP that points to the Level-3 table. This last table does not contain PTPs, but rather Page Table Entries (PTEs).

Page Table Entry

Physical Page Number	C	M	R	ACC	ET
24 bits					

Figure 4-4 Page Table Entry

Every mapped page in the system has a PTE associated with it. It is the PTE that defines the access permissions to the page (`ACC` bits), if the page was referenced (`R` bit), whether the page was modified (`M` bit) and the Physical Page Number (PPN) of the page in memory. The `ET` bits (Entry Type) identify whether the 4-byte structure is a PTP (as found in the Level-1 and Level-2 tables) or a PTE (as found in the Level-3 table) — this allows the placing of a PTE at a higher-level table if the implementor so desires. The `C` bit in the PTE determines whether a particular page is cacheable or not.

Physical Address

PPN from PTE	BO from VA
35 … 12	11 … 0

Figure 4-5 Physical Address

The low-order 12 bits of the VA are the byte offset into the page. Since a page is 4 kbytes on a Sun-4m, we need 12 bits to locate the byte in the page (remember, SPARC systems are byte-addressable: the smallest addressable memory unit is a byte — not to be confused with the smallest allocatable or mappable memory unit being 1 page). These bits actually become the low-order bits of the resulting physical memory address. The PPN from the PTE becomes the high-order 24 bits, resulting in a 36-bit physical address.

Each Level-1 table can hold a maximum of 64 PTEs; thus, we map 256 kbytes (64 x 4 kbytes-per-page) at this level. Each Level-2 table holds 64 PTPs; thus, we can point to 64 Level-3 tables, bringing our mappable address space up to 16 Mbytes (64 x 256 kbytes). Each Level-3 table can hold up to 256 PTPs, yielding a total of 4 Gbytes (256 x 16 Mb) for each process. Of course, we could have spared ourselves all that arithmetic: the maximum value that can be generated from a 32-bit unsigned integer is 4G, and a VA on SPARC is 32 bits.

As we have seen, the SRMMU references the page tables via physical addresses. The CTP register contains a physical address; the PTPs are physical addresses. As a result, SPARC (Model 120 and 140) systems, as well as the SuperSPARC without the external 1 Mb cache, must store these tables in non-cacheable memory.

So what happens when we have more processes on the system than supported contexts? Very simply, we have to use a context that already belongs to a process — thus the context number of a process must be invalidated, and the tables and pointers used by that process can now be loaded with the appropriate translation information for the new process. This means that when the process whose context we used gets switched back in to run, all the tables required to map that process's pages must be rebuilt by the OS. If we were to get into a situation where we consistently had more active processes (greater than 4096) than we had contexts, the system would spend a considerable amount of time remapping processes, and performance would degrade.

This condition, known as "context thrashing", is dealt with much more easily on the SRMMU than the old Sun MMU. Since the SRMMU places all the tables in memory, stealing a context requires only changing the root pointer in the context table and flushing the mappings from the Translation Lookaside Buffer (discussed in the next section). Under the old SRAM MMU, this required loading a lot of information into the MMU.

This is not to be confused with PMEG thrashing, which happens when the set of virtual pages currently in use spans more intermediate tables than are available. The old SRAM MMU was subject to this, since the number of PMEGS tended to be small and was not configurable. The SRMMU fixes this by placing the equivalent data structure, Level-2 and Level-3 tables, in memory. The kernel decides at boot time how many of these to allocate; thus larger installations can be configured to have more.

It seems worthwhile to illustrate the storage required to fully map 4k contexts. 4096 contexts requires 16 kbytes for the context table (plus 1536 bytes total for the three translation tables) per process. The result: 4096 contexts x 1536 bytes = 6291456. Over 6 Mb of memory would be required to store all the tables for 4096 contexts.

Translation Lookaside Buffer (TLB)

The process of translating a VA-to-PA requires locating the PTE for the page being referenced. Since the PTE lives at the last level of a 4-level table hierarchy, a little table-walking is required to locate PTEs. Obviously, if the MMU hardware could get to the PTE more quickly, the address translation process would be a lot faster and overall process execution time would improve. This is the purpose of the *Translation Lookaside Buffer* (TLB). The TLB is, very simply, a cache of PTEs. On SPARC and SuperSPARC processors, it is implemented as a 64-entry fully associative cache (that is, 64 PTEs are cached), using a LRU replacement algorithm on the SuperSPARC and a random replacement scheme on SPARC. Additionally, the root pointer, as well as the most recently used Level-2 PTP, are cached. Since the physical address of the Context Table is stored in a internal register, it is always close to the CPU, effectively cached.

The high-order bits of the Virtual Address generated by the running process, along with the context number, get used as the tag to determine if the PTE is in the TLB. Note that the context number must be used in conjunction with the VA to generate the tag, since several processes could have the same Virtual Address, but every mapped process will have a unique context number. If we get a TLB hit, the PTE data is in the hardware cache; thus the MMU can check access permissions, validity, etc., and assemble the physical address, which gets sent to the cache.

In the case of a TLB miss, the PTE must be fetched from the appropriate Level-3 table, which of course requires walking the tables, beginning at the root pointer. The actual fetching and loading of the PTE is handled by the MMU hardware, although some subtle differences exist between the SPARC and SuperSPARC implementations in this, as well as other, MMU-related areas.

The flushing and probing of the TLB at multiple levels (Context, Region, Segment, and Page) is also handled by the MMU hardware. The flush and probes are initiated via writes to a pseudo-register implemented via the alternate address space definitions provided by SPARC V8. There are assembly language routines required to do the flushes and probes. These routines are only a very small number of machine-level instructions.

A *flush* removes an entry from the TLB, and is accomplished by moving the high-order VA bits from the VA to the Flush/Probe in alternate space location, in addition to a `type` field (context, region, etc.). This data, in addition to the context number, will cause the MMU hardware to remove the appropriate PTP or PTE from the TLB.

A *probe* operation returns a PTE from either the TLB or the page table in main memory, and is initiated in a similar fashion as the flush.

Software Release Versions 5

Current Sun Software Products and Release Versions

The following tables contain lists of Sun software products and their respective release versions, current as of December 31, 1992. These tables appear monthly in the STB for your convenience.

Please contact your sales representative if you would like to update the release level of a Sun software product you already use, or wish to purchase another product. Use the tables to determine whether your release is the most current.

Please note the following software product change(s):

a. As of the release of Solaris 2.0, the C compiler will be unbundled.

Operating Systems

Product Name	Current Release
SunOS (Sun-4 architectures)	5.1
SunOS (Sun-3 architectures)	4.1.1_U1

Window Systems & Desktop Utilities

Product Name	Current Release
OpenWindows	3.0
SimplifySQL	1.1

Graphics Software

Product Name	Current Release
SunVision	1.2
XGL Graphics Library	2.0
SunGKS	4.1
SunPHIGS	2.0
SunPEX	1.0
Direct Xlib	1.0

Software Development

Product Name	Current Release
SPARCworks	2.0
SPARCompiler C *	2.0
SPARCompiler C++ *	3.0
SPARCompiler FORTRAN *	2.0.1
SPARCompiler Pascal *	3.0
Sun Ada Development Environment	1.1
Sun COBOL	1.0
Sun COBOL Runtime System	1.0
Sun COBOL Forms-2	1.0
NetISAM	1.0
OpenWindows Developer's Guide (DevGuide)	3.0.1
ONC / RPC Application Toolkit	1.0

* Available in both SPARCworks Professional and SPARCompiler versions

Software product availability is current for U.S. releases.
Availability may be delayed outside the U.S.

System Management

Product Name	Current Release
Online: Backup Copilot	1.0
Online: DiskSuite	1.0
Online: IPI Dual Port Tools	1.0
Sun SHIELD ARM and ASET	1.0
SunNet Manager	2.0
Sun Database Excelerator (SunDBE)	1.3

Other Products

Product Name	Current Release
NeWSprint	2.0
VideoPix	1.0
PC-NFS	4.0
PC-NFS LifeLine	2.0
PC-NFS Advanced Telnet	1.0
PC-NFS Programmer's Toolkit	4.0
SunPC	3.0

Software product availability is current for U.S. releases.
Availability may be delayed outside the U.S.

Connectivity Products

Product Name	Current Release
SunLink CG3270	7.1
SunLink SNA3270	7.0.1
SunLink BSC3270	6.1
SunLink BSCRJE	6.0
SunLink SNA Peer-to-Peer	7.0.1
SunLink Local 3270	6.1.1
SunLink Channel Gateway	7.1
SunLink DNI	7.0
SunLink TE320	7.0
SunLink OSI	7.0.1
SunNet MHS	7.0
SunLink IR	6.0
SunLink X.25	7.0
SunLink TRI/S	2.0
SunLink HSI/S	1.1
SunLink FDDI/S	1.0
Serial Parallel Controller	1.1
SunLink MCP	6.0
SunLink FDDI/DX	2.0
SunLink NetWare	1.0
SunLink Frame Relay	1.0
SunLink PPP	1.0

Software product availability is current for U.S. releases.
Availability may be delayed outside the U.S.

Updated Sun Operating System Support Matrix

When a platform has been designated as a final release, Sun's policy is to provide hardware and software support for out-of-production or transitioned systems and the latest compatible software for those systems for a minimum period of five years from the last order date.

As an ongoing policy, the level of support for SPARC products will generally decline at an increasing rate as new operating system releases are made available. Typically, the two most current versions of the operating system will be supported at level 1.

The details of Sun's software support policy have been published in a Sun Operating System Support matrix in past STB issues. As illustrated in the support matrix, older versions of the operating system generally receive lower levels of software support than newer versions. Thus, we recommend that you upgrade your system to the most current versions to ensure you receive the best quality of support from Sun.

Since we continually enhance the operating system to meet our customers' needs, we promised to keep you abreast of updates to the support matrix via the STB. Now that SunOS 4.1.3 and Solaris 2.0 are available, we provide an updated matrix.

Sun uses "support levels" to describe its software support policy. Please note that the services described in these levels are subject to local availability—contact your local service manager should you have any questions.

Level 1 Support

Level 1 offers the complete array of services contained in Sun's Personal AnswerLine Service, AnswerLine Service, and Software Update Service (see Note ①). Level 1 support also includes the full availability of options such as SunStart™, Sun's educational services, and supplementary consulting products and services.

Level 2 Support

Level 2 includes Sun's Personal AnswerLine and AnswerLine levels of phone support, limited availability of software and documentation, bug report submission and tracking, and emergency fixes and workarounds (see Note ①). Level 2 support also includes availability of supplementary consulting products and services.

Level 3 Support

Level 3 provides Time and Materials (T&M) support only.

Note ① — Software updates will only be available for the most current release. When a version of software has been designated as the final release for a platform, only critical bug fix releases will be available for that platform.

Sun Operating System Software Support Matrix

The following tables illustrate Sun's OS software support matrix. See Note ② below regarding this section.

For Sun-2	3.5	4.0/4.0.1	4.0.3	(see Note ③)
1/91-1/92	level 3	level 3	level 2	
1/92-1/94	level 3	level 3	level 3	

For Sun-3	3.5	4.0/4.0.1	4.0.3/4.0.3e	4.1	4.1.1	(see Note ③)
1/91-1/92	level 3	level 3	level 2	level 1	level 1	
1/92-1/95	level 3	level 3	level 3	level 2	level 2	
1/95-1/96	level 3	level 3	level 3	level 3	level 3	

For Sun386i	4.0.1i	4.0.2i	(see Note ③)
1/91-1/92	level 2	level 1	
1/92-1/95	level 3	level 2	
1/95-1/96	level 3	level 3	

For SPARC	3.5	4.0/4.0.1	4.0.3/4.0.3e	4.1	4.1.1	4.1.2	Solaris 2.0	4.1.3
1/91-present	level 3	level 3	level 3	level 2	level 2	level 1	level 1	level 1

Note ② — The above levels 1 and 2 are for contract customers only. Non-contract customers have the option to purchase support on a Time and Materials basis or use Sun's consulting services. The then-current unbundled products will be supported at the same level as the customer's operating system. Support for emergency fixes and workarounds may require the customer to upgrade to a more current version of the product.

Note ③ — These will be the final releases for these product lines, with the exception of critical bug fix releases. Sun's future SVR4-based operating system will function primarily on SPARC processor-based systems (Intel-based support for SVR4 is available with SunSoft's INTERACTIVE UNIX product line). Interoperability between Sun-3 or Sun386i systems and SPARC SVR4 systems will be fully supported through services such as NFS, NIS, remote login, and the like.

In addition, Sun is committed to supporting Sun-3 diskless clients running SunOS 4.1.1 on Solaris 2.0 servers.

Notes & Comments 6

World Hotlines

Sun Customers throughout the world have service hotlines available for both software and hardware support questions. The service hotlines are shown below. If your country is not shown in the table, please phone your local Sun sales office. World hotlines are divided into those for the U.S.A., Europe, and Intercon.

United States

All	**1-800-USA-4SUN** (**1-800-872-4786**)

Europe

Finland	Sun Microsystems Oy	**+358-0-502-2700**
France	Sun Microsystems France SA	**+33-1-30-67-50-80**
Germany	Sun Microsystems GmbH	**+49-89-46008-321**
Italy	Sun Microsystems Italia SpA	**+39-6055228**
Netherlands	Sun Microsystems Nederland BV	**+33-501234**
Spain	Sun Microsystems Iberica SA	**+34-1-5560243**
Sweden	Sun Microsystems AB	**+46-8-623-9100**
Switzerland	Sun Microsystems (Schweiz) AG	**+41-1-825-7272**
	Sun Microsystems (Suisse) SA	**+41-22-364-3639**
United Kingdom		
	Sun Microsystems UK Ltd	**+44-276-691974**

Intercon

Canada:

Quebec District	Montreal, Quebec	**(514) 744-9240**, **(800) 361-1554**
Ottawa District	Ottawa, Eastern Canada	**(613) 723-8112**, **(800) 465-4415**
Central District	Toronto, Central Ontario	**(416) 490-9490**, **(800) 263-1680**
Prairies District	Manitoba, Saskatchewan, Alberta, Calgary, Western Canada	**(403) 262-6722**, **(800) 661-9256**
Edmonton District	Edmonton	**(403) 421-7264**, **(800) 661-3643**
B. C. District	Vancouver, British Columbia	**(604) 684-4120**, **(800) 663-0440**
	National CSS/Education	**(416) 477-6745**

Australia	Sun Australia	**(008) 02-4417**
New Zealand	Auckland	**64-9-525-2749**
	Wellington	**64-4-499-2344**
Hong Kong	Sun Asia	**852-802-4188**
Japan	Nihon Sun	**044-819-3121**

Latin America: Contact the Latin American hotline at ICON headquarters:
U.S. Office **(415) 688-9160**

Within ***Brazil***, you may also contact:
Sun do Brasil **55-11-887-9011**

Caribbean: Call your local distributor.

All Other Countries
Call **(415) 688-9372**, which is a U.S. number.

Reporting Bugs

This article contains three sections for submitting bugs. The first section describes procedures to use within the United States and Canada. The second section describes procedures within Europe. The third section describes procedures within ICON.

Submitting Software Bugs: United States and Canada

This section contains information on reporting bugs within the U.S., for customers holding and not holding support contracts.

Sun's United States Solution Center accepts software bug reports from Sun users via electronic mail and by phone. The method you use to submit a bug report varies with your needs.

U.S. users holding support contracts can report bugs to US Solution Center via the **(800) USA-4-SUN** phone hotline. Canadian users holding support contracts should call their local support center. The U.S. Solution Center phone hotline is the fastest way for a customer to find out if a problem is known and if a workaround exists. The status of previously-reported bugs can also be obtained in this way.

U.S. customers holding support contracts can also submit bug reports electronically to the address *sun!hotline* (or *hotline@Sun.COM*). Canadian customers need to use *hotline@Canada.Sun.COM*. This method generates a service order, and can be used when lines of code or other information difficult to relay over the phone is needed to describe the bug.

Customers who do not hold Sun software support contracts can report bugs via electronic mail to the address *sun!sunbugs* (or *sunbugs@Sun.COM*). These reports are reviewed periodically to determine proper disposition. Those reports determined to be from supported customers are forwarded to the U.S. Solution Center for handling. Reports from customers who cannot be verified as holding a support contract are reviewed by Sun's engineering and support personnel. An internal bug report is generated if the reported bug is new and verifiable.

Finally, customers not holding software support contracts may call the **(800) USA-4-SUN** phone hotline to report a problem and request support on a Time and Materials (T&M) basis. Canadian customers should call their local support center. In this case, please have a Purchase Order (PO) number for billing purposes.

Summary: United States and Canada

For U.S. contract customers, **(800) USA-4-SUN** is the best method to report bugs. Canadian contract customers should report bugs to their local support center. The hotlines to use appear monthly in the note entitled "World Hotlines."

The electronic mail address *sun!hotline* (or *hotline@Sun.COM*) is available for U.S. customers to submit materials that are difficult to relay over the phone. Canadian customers should use *hotline@Canada.Sun.COM* .

For non-contract customers, the electronic mail address *sun!sunbugs* (or *sunbugs@Sun.COM*) is available to report bugs. To help us serve you better, please include the following information with all electronic mail reports:

- Your name
- The name and address of your organization
- Your Sun site code, if available
- Your workstation model and serial number
- The software release(s) you are running
- A description of the problem that you are experiencing

Submitting Software Bugs: Europe

This section contains information on reporting bugs within Europe, for customers holding and not holding support contracts.

Procedures for submitting bugs are similar to those used in the United States. All customers should use their local country Answer Center to report bugs, with contract customers receiving a specific follow-up.

Sun customers not holding software service contracts can call their local Answer Center, and will need to provide a Purchase Order (PO) number at the time of the call.

Summary: Europe

To help European service centers serve you better, please include the following information with all electronic mail reports:

- Your name
- The name and address of your organization
- Your Sun site code, if available
- Your workstation model and serial number
- The software release(s) you are running
- A description of the problem that you are experiencing

Finland

Report bugs to the Finland Answer Center at the postal address shown below.

Hotline	Hotline Telephone: +358-0-502-2700
Sun Microsystems Oy	Telefax: +358-0-502-2800
Sinikalliontie 12	
SF-02630 Espoo, Finland	E-mail Address: *Hotline@Finland.Sun.COM*

Special Dispatch Arrangements: Provide Dispatch with the Contract number.
Arrangements for Non-Contract Customers: Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

France

Report bugs to the France Answer Center at the postal address shown below.

Service Hot Line	Hotline Telephone: +33-1-30-67-50-80
Sun Microsystems France SA	Telefax: +33-1-30-67-53-70
13, Avenue Morane Saulnier	Minitel: 36 14 code SUNFRANCE
BP 62	
78142 Velizy, France	

Special Dispatch Arrangements: Please provide Dispatch with System serial number *or* Contract number.
Arrangements for Non-Contract Customers: Please provide a valid PO number for billing on a Time and Materials (T&M) basis, and order this support at the above address.

Germany

Report bugs to the Germany Answer Center at the postal address shown below.

Hotline	Hotline Telephone: +49-89-46008-321
Sun Microsystems	Telex: 5218197 sun
Bretonischer Ring 3	Telefax: +49-89-46008-320
8011 Grasbrunn, Germany	E-mail Address:
	{*sunuk,unido*}*!sunde!hotline* or
	hotline@Germany.Sun.COM

Arrangements for Non-Contract Customers: Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

Italy

Report bugs to the Italy Answer Center at the postal address shown below.

Hotline Sun Microsystems Italia SpA Centro Direzionale Colleoni Palazzo Andromeda (Ingresso 1) Via Paracelso 16 20041 Agrate Brianza Milan, Italy	Telephone: +039-6055228 Telex: 326423 MBOX SUNITA Telefax: +039-6056764

Arrangements for Non-Contract Customers: Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

The Netherlands

Report bugs to The Netherlands Answer Center at the postal address shown below.

Sun Microsystems Nederland BV Computerweg 1 3821 AA Amersfoort Netherlands	Hotline Telephone: +33-501234 Telefax: +33-553058 Telex: 79297 sun nl

Arrangements for Non-Contract Customers: Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

Spain

Report bugs to Spain Answer Center at the postal address shown below.

Sun Microsystems Iberica SA Plaza Pablo Ruiz Picasso s/n Torre Picasso Planta 25 28020 Madrid, Spain	Hotline Telephone: +34-1-5560243 Telefax: +34-1-5564097

Arrangements for Non-Contract Customers: Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

Sweden

Report bugs to the Sweden Answer Center at the postal address shown below.

Sun Microsystems AB
Box 51
S-164 94
Kista, Sweden

Hotline Telephone: +46-8-623-9100

E-mail Address: *hotline@Sun.se* or *hotline@Sweden.Sun.COM*

Arrangements for Non-Contract Customers:
Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

Switzerland

Report bugs to the Switzerland Answer Center at one of the postal addresses shown below.

(*German*)
Sun Microsystems (Schweiz) AG
Postfach
Eschenstrasse 8
CH-8603 Schwerzenbach, Switzerland

Telephone: +41-1-825-7272
Telefax: +41-1-825-7222
E-mail: *hotline@Swiss.Sun.COM*

(*French*)
Sun Microsystems (Suisse) SA
Route des Avouillons 12
CH-1196 Gland, Switzerland

Telephone: +41-22-364-3639
Telefax: +41-22-364-1176
E-mail: *answerline@Swiss.Sun.COM*

Special Dispatch Arrangements: Provide Dispatch with the Contract number and System serial number.
Arrangements for Non-Contract Customers: Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

United Kingdom

Report bugs to the UK Answer Center at the postal address shown below.

Sun Microsystems UK Ltd
Watchmoor Park
Riverside Way
Camberley, Surrey
GU15 3YL, England

Hotline Telephone: 0276-691974
Telefax: 0276-681546

Special Dispatch Arrangements: Please provide Dispatch with either the System serial number *or* Contract number.
Arrangements for Non-Contract Customers: Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

Submitting Software Bugs: Intercon

This section contains information on reporting bugs within Intercon, for customers holding and not holding support contracts.

Procedures for submitting bugs are similar to those used in the United States. All customers should use their local country Answer Center to report bugs, with contract customers receiving a specific follow-up.

Sun customers not holding software service contracts can call their local Answer Center, and will need to provide a Purchase Order (PO) number at the time of the call.

Summary: Intercon

To help Intercon service centers serve you better, please include the following information with all electronic mail reports:

- Your name
- The name and address of your organization
- Your Sun site code, if available
- Your workstation model and serial number
- The software release(s) you are running
- A description of the problem that you are experiencing

Detailed information for individual countries follows.

Australia

Report bugs to the Australian Answer Center at the postal address shown below.

Hotline	Hotline Telephone: (008) 02-4417
Sun Microsystems Australia Pty. Ltd.	Telefax: 02 436 1084
PO Box 320	E-mail Address: *hotline@Aus.Sun.COM*
Artarmon, New South Wales 2064	
Australia	

Special Dispatch Arrangements: Please provide Dispatch with either the System serial number *or* Contract number.

Arrangements for Non-Contract Customers: Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

Japan

Report bugs to the Nihon Sun Answer Center at the postal address shown below.

Nihon Sun Microsystems K.K	Hotline Telephone:
KSP Technical Center - Answer Center	044-819-3121, 044-819-3111
KSP R&D Bldg. B-1F-112	FAX: 044-819-3120
Sakato 100-1, Takatsu-ku,	
Kawasaki, Kanagawa, 213	
Japan	

Special Dispatch Arrangements: Provide Dispatch with the Contract number and Serial Number.

Arrangements for Non-Contract Customers: Please provide a valid PO number for billing on a Time and Materials (T&M) basis.

New Zealand

Report bugs to the New Zealand Answer Center at the number shown below.

Hotline Telephone: 4-499-2344

Your STB Subscription

Your company's software service contract includes a monthly issue of the STB. Each month, the copy of your STB is mailed to your company's primary contact person or department named in the contract. Sites with more than one contract may receive more than one STB copy, depending on how the contracts are arranged.

Duplicating Your STB

Your primary contact person or department may duplicate this "master" STB copy for all Sun workstation end-users. As long as you duplicate copies and route them internally only, there are no copyright infringement problems.

This limited permission for duplication is for your convenience only, however, and does not include any duplication for resale, for distribution outside your company, or for distribution to employees of companies not having a Sun software service contract.

Multiple STB Subscriptions

You can now order extra copies of the STB for a low monthly fee for each additional copy. These extra copies will sent to the same address as your "master" STB copy. Please contact your local Sales/Service Office or Sun Authorized Distributor.

Changes or Updates

Changes of address or other updates, including change of contact name, should be submitted to your Customer Service Administrator (CSA), not to the STB Editor. Your CSA will register the requested changes through proper channels with Sun headquarters.

Further STB Questions

If you have any questions or comments regarding the STB, including corrections, please send your comments to *stb-editor@Sun.COM* or *sun!stb-editor*.

Index

A

Andrew File System (AFS), 3-2, 3-3, 3-4, 3-5, 3-8
automounter, 3-5, 3-12, 3-14

C

Cell Directory Service (CDS), 2-10, 2-12

D

Data Link Provider Interface (DLPI), 2-15
DECnet, 2-18
Device Driver Interface (DDI), 2-15
Device Kernel Interface (DKI), 2-15
Distributed Computing Environment (DCE), 2-1, 2-5, 2-9, 2-12, 2-13, 2-18, 3-2, 3-5
Distributed Objects Everywhere (DOE), 2-14
Domain Name Service (DNS), 3-5

E

Ethernet, 3-2 to 3-3, 3-4, 3-8, 3-9, 3-10
External Data Representation (XDR), 2-18

F

FDDI, 3-3, 3-9, 3-10
Federated Services, 2-1 to 2-18
    Federated Service Interfaces (FSI), 2-1, 2-7

G

GX, 3-16

I

Interface Definition Language (IDL), 2-14
ISDN (Integrated Services Digital Network), 2-16, 3-10

K

Kerberos, 2-13, 2-17, 3-13, 3-14

M

Mbus, 4-2
Memory Management Unit (MMU), 4-1, 4-3, 4-4, 4-8, 4-9, 4-10

N

NetWare, 2-1, 2-5, 2-17
NetWare SunLink, 2-18

Network File System (NFS), 2-2, 2-8, 2-13, 2-17, 3-1 to 3-14, 3-15
Network Lock Manager (NLM), 3-8
NIS (Network Information Service), 2-17, 3-5
NIS+ (Network Information Service Plus), 2-9, 2-10, 2-13, 2-17, 3-5, 3-14

O

ONC, 2-2, 2-8, 2-9, 2-13, 2-14, 2-17
ONC+, 2-1, 2-5, 2-7, 2-17 to 2-18
Online DiskSuite, 3-11
Open Software Foundation (OSF), 2-5, 2-10, 2-13, 2-18, 3-2
Open Systems Interconnection (OSI), 2-1, 2-10, 2-13, 2-15, 2-18

P

page map entry groups (PMEGs), 4-4, 4-8
page table entry (PTE), 4-3, 4-4, 4-7 to 4-8, 4-9 to 4-10
page table pointer (PTP), 4-7, 4-8, 4-9, 4-10
physical page number (PPN), 4-7

R

Remote Procedure Calls (RPC), 2-13 to 2-14, 3-1, 3-10, 3-13, 3-14
RISC (Reduced Instruction Set Computing), 2-18

S

Secure NFS, 3-13
Secure RPC, 3-13, 3-14
SNA, 2-18
Solaris, 2-1 to 2-18
SPARC, 4-1 to 4-10
SPARC Reference Memory Management Unit (SRMMU), 4-1, 4-2, 4-3 to 4-5, 4-8
SPARCstation, 3-15
STREAMS, 2-8, 2-15
SuperSPARC, 4-2, 4-5, 4-8, 4-9
System V Release 4 (SVR4), 2-8, 2-15, 2-17

T

TCP, 2-15, 3-2, 3-9, 3-10
TCP/IP, 3-2
ToolTalk, 2-14
Translation Lookaside Buffer (TLB), 4-8, 4-9 to 4-10
Transport Layer Interface (TLI), 2-15
Transport Provider Interface (TPI), 2-15
Transport-Independent Remote Procedure Call (TI-RPC), 2-13, 2-15, 2-18

U

UDP, 3-9
Unix File System (UFS), 2-8

V

vnode, 2-8 to 2-9, 4-4

X

X.500 Directory Services, 2-10, 2-12
X/Open Transport Interface (XTI), 2-15